UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

SISTEMAS DE INFORMAÇÃO

FUNDAMENTOS DE COMPILADORES

COMPILADOR Cmm

Trabalho apresentado como parte das avaliações parciais da disciplina Fundamentos de Compiladores do curso de Sistemas de Informação do Campus I da UNEB.

JOÃO PEDRO HEGOUET
JULIANA DE CARVALHO SANTANA

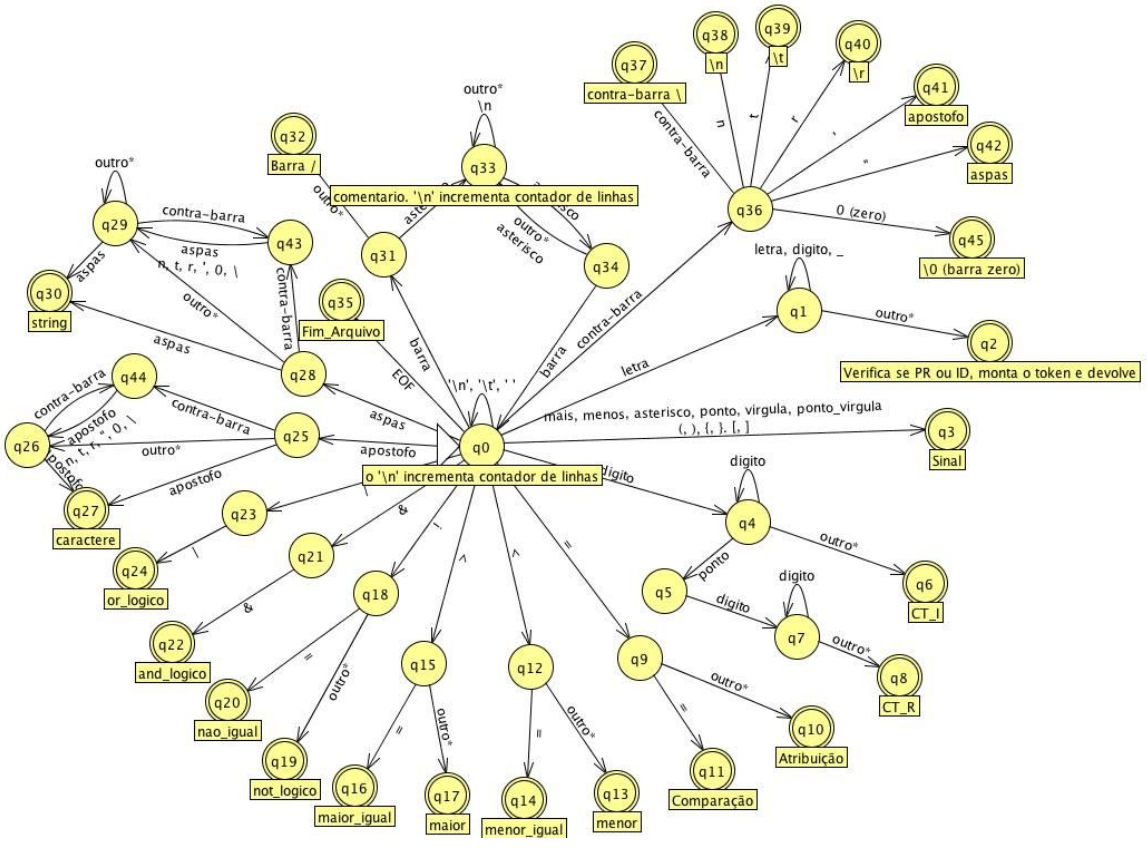
2017.2

1. Introdução

O projeto do Compilador Cmm foi dividido em 3 fases. A primeira fase consistiu em criar um AFD para a análise léxica e, baseado nele, desenvolver e implementar o analisador léxico do compilador.
Após a conclusão dessa etapa, foi desenvolvida a análise sintática do compilador, o gerenciador da tabela de símbolos e o gerenciador de erros finalizando, assim, a segunda fase do projeto.
Por fim, a terceira fase do projeto é caracterizada pela implementação da análise semântica do compilador e o gerador de código da máquina de pilha.
A seguir, será detalhado cada processo separadamente, indicando as principais decisões, as principais estruturas utilizadas, o que foi e o que não foi implementado, além de outras informações.

2. Análise Léxica

O AFD a seguir foi criado baseando-se nas especificações da linguagem cmm e utilizado como base para o desenvolvimento do analisador léxico.



No analisador léxico foram declarados um vetor de **categorias** com as possíveis categorias para os tokens (palavra reservada, sinal, identificador, inteiro, real, caracter, cadeia de caracter e fim

de arquivo), um vetor do tipo **token** com os possíveis **sinais** e **operadores** e um conjunto de palavras reservadas da linguagem chamado **PR**.
O analisador foi desenvolvido baseado no AFD e todas as regras léxicas propostas pela linguagem foram implementadas. Para desenvolvê-las foi criado um tipo de estrutura chamado **token**, que possui atributos que caracterizam cada token.
//Foram criadas as funções **ehsinal**, **copiaTokenSinais**, **transicao_sinais** e, a mais importante, **analex** que faz a análise letra por letra do código e verifica-o de acordo com as regras léxicas.

3. Análise Sintática

No analisador sintático foram desenvolvidas 8 funções que determinam as regras de produção da gramática. As principais são a **prog** e a **cmd**.
A função **prog** é aquela que verifica as declarações de variáveis e de funções e, também, a inicialização das funções. É nela que armazenamos os identificadores na tabela de símbolos quando os encontramos. A função **cmd** é a função que verifica os comandos da linguagem no código.
Também foram desenvolvidas as funções **atrib** que verifica as atribuições do código, **op_rel** que verifica se é um operador relacional, **Expressao** que verifica a ocorrência de expressões simples ou de comparações entre elas, **expr_simp** que verifica a formação de expressões simples, **Termo** que verifica a ocorrência de fatores ou operações entre fatores e a função **Fator** que procura no código identificadores, inteiros, reais, caracteres e cadeias de caracteres.

4. Análise Semântica

O analisador semântico foi desenvolvido baseado nas regras semânticas propostas pela linguagem. Foram implementadas as regras semânticas que diz a respeito das **declarações** e aos **requisitos de consistência de tipos** na definição de funções, nas expressões e nos comandos.

5. Gerenciador de Tabela de Símbolos e Gerenciador de Erros

Os símbolos são adicionados na tabela de símbolos todas as vezes que é encontrado um identificador no código. Foi declarado para o gerenciador de símbolos um tipo de estrutura **simbolo**, um tipo **escopo** que indica se é global ou local e implementadas as funcões **apagasSimbolos** e **existeID**.

O gerenciador de erros é o arquivo que contém os tipos de erros que podem ocorrer durante a compilação do código. Ele possui um vetor de cadeias de caracteres que descrevem erros de validação, má formação de expressões, sinais e identificadores que eram esperados, quantidade de parâmetros de uma função errada, conflitos entre tipos e muitos outros. Neste arquivo foi

implementada uma função chamada **erro** que recebe um inteiro com o código do erro e imprime o erro na tela, informando a linha em que o erro ocorreu e finalizando o programa.

6. Gerador de Código da MP

O gerador de código da máquina de pilha foi desenvolvido de acordo com as especificações dadas pelo professor. Foram implementadas as funções de manipulação de dados, de instruções com valores inteiros, reais, de controle de fluxo de início e fim de programa.

7. Conclusão

O desenvolvimento do projeto e o estudo da funcionalidade do compilador influenciou diretamente na nossa formação como estudantes da área de computação, mais especificamente, de Sistemas de Informações. Conhecendo o funcionamento do compilador, nos ajuda a implementar melhor nossas soluções