**Geração de Código Intermediário**

**Mateus Tomoo Yonemoto Peixoto**

DACOM – Universidade Tecnológica Federal do Paraná (UTFPR)

Caixa Postal 271 – 87301-899 – Campo Mourão – PR – Brazil

{mateustomoo}@gmail.com

**Abstract.** This paper describes the development of the last parto f the discipline implementation work, intermediate code generation for a compiler being designed for the T++ programming language.

**Resumo.** Este artigo descreve o desenvolvimento da última parte do trabalho de implementação da disciplina, a geração de código intermediário para um compilador projetado para a linguagem de programação T++.

# Introdução

Um compilador exige alguns passos até que um código intermediário seja obtido. Esses passos são divididos em análise léxica, análise sintática, análise semântica e geração de código.

A Geração de Código é a fase final de um compilador e também a fase mais complexa, pois não depende exclusivamente da linguagem-fonte, mas também de informações e detalhes da arquitetura-alvo, da estrutura do ambiente de execução e do Sistema Operacional da máquina-alvo.

Para o desenvolvimento deste trabalho, foi utilizado a linguagem de programação Python (versão 3.6.2) e o llvmlite.

# Geração de Código Intermediário

Uma estrutura de dados que represente o programa-fonte durante a tradução é denominada Representação Intermediária (IR).

Chamamos de código intermediário a forma de representação intermediária gerada a partir da árvore sintática que se assemelhe melhor ao código-alvo. O código intermediário pode assumir muitas formas, porém todas representam a linearização da árvore sintática (representação sequencial da árvore sintática). Nele também pode conter informações detalhadas sobre a máquina-alvo e o ambiente de execução.

# LLVM – IR

Os principais componentes de um código LLVM-IR são:

* Módulo – representa um arquivo com código-fonte ou uma unidade de tradução, onde todo o restante do código deve estar dentro de um módulo.
* Funções – módulos contém funções, que são declaradas com seus nomes e argumentos. Uma função é um container de Basic Blocks.
* Basic Block (bloco básico) – pedaço contíguo de instruções.
* Instruções – operação única expressa em um código. Operações como adição, subtração ou até mesmo uma instrução de load ou store.

# Implementação

# Referências

LOUDEN, Kenneth C. Compiladores: princípios e práticas. São Paulo, SP:

Thomson, c2004. xiv, 569 p. ISBN 8522104220.

http://hackingoff.com/compilers/regular-expression-to-nfa-dfa

PLY (Python Lex-Yacc). Disponível em: <http://www.dabeaz.com/ply/>.