Compiladores – 2016/2
Roteiro de Laboratório 11

1 Objetivos

O objetivo deste laboratório é implementar um interpretador de código para programas escritos na linguagem TINY.

2 Representação intermediária

Um interpretador depende de uma representação intermediária (*intermediate representation* – IR) do código de entrada. Nesse laboratório vamos usar como IR uma **Árvore de Sintaxe Abstrata** – ***Abstract Syntax Tree (AST)***. Os nós de uma AST correspondem aos comandos da linguagem e os filhos de um nó são as dependências do comando.

Uma AST pode ser vista como uma versão simplificada da *parse tree* aonde somente as informações necessárias para a execução do programa são mantidas. O ponto chave para a construção da AST é definir quais informações devem ficar guardadas na árvore. Uma vez que isso é feito, a implementação do executor de código deve seguir a mesma especificação.

Os arquivos de *template* disponibilizados no AVA realizam a construção da AST. Faça o *download* do código e entenda o seu funcionamento. A tarefa deste laboratório é desenvolver o código do arquivo `interpreter.c`.

3 Executor de código

O executor de código (interpretador) é um programa (função `run_ast`) que recebe como entrada uma AST e caminha na árvore, realizando as operações indicadas nos nós da AST. Para o armazenamento de resultados temporários é utilizada uma **pilha**. Desta forma, o executor trata a AST como uma IR de um endereço (*one-address IR*).

O caminhamento do executor pela AST deve ser em **pós-ordem**, devido às dependências de cada operação. Ao terminar a execução de uma dada operação, o resultado fica armazenado na pilha, para ser usado depois. Por exemplo, uma operação de soma de inteiros desempilha os dois primeiros valores da pilha, realiza a operação e empilha de volta o resultado.

O caminhamento pela AST pode ser implementado tanto de forma recursiva quanto de forma iterativa, como você preferir. Além da pilha, o executor também possui uma área de memória para o armazenamento dos valores das variáveis do programa. Existem duas formas de implementar essa área de memória:

1. Estender a tabela de símbolos para incluir um campo adicional para o valor corrente da variável. Isso só é possível porque a linguagem TINY é tão simples que permite a alocação estática das variáveis.

2. Criar uma estrutura que mapeia variáveis em endereços de memória (um índice em um vetor de inteiros).

4 Passos para realização do laboratório

1. Faça o *download* dos arquivos de *template* no AVA.
2. Implemente a função `run_ast`. Use as figuras das ASTs fornecidas pelo professor para inferir o funcionamento do seu executor.
3. Teste o seu interpretador. Os programas de entrada são os mesmos dos laboratórios anteriores.

Observações importantes:

- A linguagem TINY tratada aqui é a especificação original, isto é, sem as extensões descritas no Laboratório 09.
- Uma implementação de referência para esse laboratório será disponibilizado pelo professor em um futuro próximo. No entanto, você é *fortemente* encorajado a realizar a sua implementação completa antes de ver uma solução em outro lugar.