Processamento/Teoria de Linguagens e Compilação
LCC (3ºano) + MEFis (1ºano)

Trabalho Prático nº 2 (Gramáticas, Compiladores)

Ano lectivo 21/22

Objectivos e Organização

Este trabalho prático tem como principais **objectivos**:

- (genericamente) aumentar a experiência em *engenharia de linguagens* e em *programação generativa (gramatical)*, reforçando a capacidade de escrever gramáticas, quer independentes de contexto (GIC), quer tradutoras (GT);
- (especificamente) desenvolver processadores de linguagens segundo o método da *tradução dirigida pela sintaxe*, a partir de uma gramática tradutora;
- (especificamente) desenvolver um **compilador** gerando código para uma **máquina de stack virtual**, no ano corrente será usada a VM, Virtual Machine[1];
- (especificamente) utilizar *geradores de compiladores* baseados em *gramáticas tradutoras*, concretamente o Yacc, versão PLY do Python, completado pelo *gerador de analisadores léxicos* Lex, também versão PLY do Python;

e como **objectivos** secundários:

- rever e aumentar a capacidade de escrever *gramáticas independentes de contexto* que satisfaçam a condição LR() usando BNF-puro
- criar o hábito de escrever a documentação (os relatórios dos trabalhos práticos e projectos) em LaTeX.

Para o efeito, esta folha contém apenas 1 enunciado.
O programa desenvolvido será apresentado aos membros da equipa docente, totalmente pronto e a funcionar (acompanhado do respectivo relatório de desenvolvimento) e será defendido por todos os elementos do grupo.
O **relatório** a elaborar, deve ser claro e, além do respectivo enunciado, da descrição do problema, das decisões que lideraram o desenho da linguagem/gramática e as regras de tradução para Assembly da VM, deverá conter exemplos de utilização (programas-fonte diversos e respectivo código produzido). Como é de tradição, o relatório será escrito em LaTeX.

[1] Manual da Máquina e Simulador, para interpretar o Assembly gerado pelo compilador, já disponíveis no Blackboard.

1 Enunciado

Pretende-se que comece por definir uma linguagem de programação imperativa simples, a seu gosto.
Apenas deve ter em consideração que essa linguagem terá de permitir:

- *declarar* variáveis atómicas do tipo *inteiro*, com os quais se podem realizar as habituais operações aritméticas, relacionais e lógicas.
- *efetuar* instruções algorítmicas básicas como a *atribuição do valor de expressões numéricas a variáveis.*
- *ler* do *standard input* e *escrever* no *standard output.*
- *efetuar* instruções *condicionais* para controlo do fluxo de execução.
- *efetuar* instruções *cíclicas* para controlo do fluxo de execução, permitindo o seu aninhamento.
  Note que deve implementar pelo menos o ciclo `while-do`, `repeat-until` ou `for-do`.

Adicionalmente deve ainda suportar, à sua escolha, uma das duas funcionalidades seguintes:

- *declarar e manusear* variáveis estruturadas do tipo *array (a 1 ou 2 dimensões) de inteiros*, em relação aos quais é apenas permitida a operação de indexação (índice inteiro).
- *definir e invocar subprogramas* sem parâmetros mas que possam retornar um resultado do tipo inteiro.

Como é da praxe neste tipo de linguagens, as variáveis deverão ser declaradas no início do programa e não pode haver re-declarações, nem utilizações sem declaração prévia. Se nada for explicitado, o valor da variável após a declaração é 0 (zero).
Desenvolva, então, um compilador para essa linguagem com base na GIC criada acima e com recurso aos módulos Yacc/ Lex do PLY/Python.
O compilador deve gerar **pseudo-código**, Assembly da Máquina Virtual VM.
Muito Importante:
Para a entrega do TP deve preparar um conjunto de testes (programas-fonte escritos na sua linguagem) e mostrar o código Assembly gerado bem como o programa a correr na máquina virtual VM. Esse conjunto terá de conter, no mínimo, os 4 primeiros exemplos abaixo e um dos 2 últimos conforme a sua escolha acima:

- ler 4 números e dizer se podem ser os lados de um quadrado.
- ler um inteiro N, depois ler N números e escrever o menor deles.
- ler N (constante do programa) números e calcular e imprimir o seu produtório.
- contar e imprimir os números impares de uma sequência de números naturais.
- ler e armazenar N números num array; imprimir os valores por ordem inversa.
- invocar e usar num programa seu uma função 'potencia()', que começa por ler do input a base B e o expoente E e retorna o valor B^E.