Processamento de Linguagens e Compiladores
(3º ano de LCC)

Galo

TP3

Grupo 14

Artur Queiroz	Rafael Fernandes	Rafaela Pinho
A77136	A78242	A77293

14 de Janeiro de 2018

Resumo

Neste relatório apresentamos a linguagem que criamos e o complidor que gera o código para a Máquina Virtual VM.

Conteúdo

Capítulo 1

Introdução

Capítulo 2

Galo e Compilador

2.1 Descrição informal do problema

Neste trabalho foi pedido para criarmos uma linguagem de programação imperativa e desenvolver um compilador para a linguagem criada.

Na linguagem as declarações de variáveis devem ser colocadas no início do programa, não pode haver re-declarações e não se pode usar variáveis sem estar declaradas primeiro. Caso não seja atribuido uma valor à variável depois da declaração, esta ficará com o valor zero.

O complilador deve gerar o código assembly para a Máquina Virtual VM.

2.2 Especificação dos requisitos

Para este tarbalho a linguagem que criamos tem de conter os seguintes requisitos:

1) Declarar e manusear variáveis atómicas do tipo inteiro e estruturas do tipo array de inteiros.

2) Ler do standard input e escrever no standard output.

3) Fazer instruções básicas como a atribuição de expressões a variáveis.

4) definir e invocar subprogramas sem parâmetros mas que possam retornar um resultado atómico(?)

5) efetuar instruções para controlo do fluxo de execução—condicional e cíclica—que possam ser aninhadas. (?)

2.3 Expressões regulares

As expressões regulares usadas foram:

1)

2.4 A nossa linguagem

2.4.1 Galo

Como já referido em cima, foi-nos pedidos para criar uma linguagem de programação. Decidimos chamar de Galo por ser um símbolo típico de Potugal, e atribuimos .gl para a extensão.

Para a definirmos utilizamos uma gramática independente do contexto, em que tomamos certas decisões que serão especificadas.

O Galo reconhece os segintes tipos: números inteiros (int), números décimais (float) e conjunto de caractéres (string).

Capítulo 3

Codificação e Testes

3.1 Problemas de implementação, Decisões e Alternativas

3.1.1 Problemas de implementação

3.1.2 Decisões

1) O E (&&) está definida pela multuplicação e o OU (||) pela adição.

Tabela 3.1: Tabela do E

*(E)	0	1
0	0	0
1	0	1

Tabela 3.2: Tabela do OU

+(OU)	0	1
0	0	1
1	1	2

2) Não se pode declarar mais do que uma variável numa linha.
Exemplo:
int a = 2 , c = 0;
terá de ser:
int a = 2;
int c = 0;

3) Não se pode fazer "return"dentro dos Se's e dos Enq's.

4) Nas expressões numéricas, as operações binárias têm de estar sempre dentro de parênteses.

5) O Se tem de ter sempre Senao.

3.1.3 Alternativas

3.2 Testes realizados e Resultados

Capítulo 4

Conclusão

Hoje em dia já existem muitos pré-processadores que facilitam a escrita de documentos em HTML.
Tendo em conta os aspetos apresentados ao longo do relatório, conclui-se que o *Flex* é uma boa ferramenta para fazer o pré-processador e que com ele se torna fácil programar usando expressões regulares e um pouco de linguagem C.
Como trabalho futuro poderíamos acrescentar mais símbolos para completar o pré processador.