

# **Relatório Trabalho Prático**

(Compilador em OCaml da linguagem Arith para Assembly MIPS)

**Trabalho realizado por:**

André Martins, a37413

João Brito, a37880

# Índice

1. Introdução (*pág.* 3)
2. A linguagem Arith (*págs.* 4 – 8)

## **Introdução**

O presente relatório complementa o trabalho prático desenvolvido no âmbito da Unidade Curricular de Processamento de Linguagens (cód.11567).

Tratará de apresentar a linguagem Arith, nos seus moldes teóricos, bem como na sua aplicação prática. Adicionalmente, fornecerá documentação apropriada do código desenvolvido e um manual de utilizador intuitivo.

# A Linguagem Arith

Sendo uma linguagem aritmética por natureza, a sua sintaxe assenta em variáveis, constantes, expressões e instruções.

Podemos declarar/atribuir valores a variáveis:

`var = e`

Bem como, imprimir o resultado de expressões no ecrã:

`print e`

A **sintaxe abstrata** desta linguagem caracteriza-se da seguinte forma:

## Expressões

`e ::=`  
| `const (n : int ou b : boolean)`  
| `var (identificador)`  
| `e op e (+, -, *, /)`  
| `( e )`  
| `set var = e1 in e2`

## Instruções

$i ::=$

- | `var = e`
- | `print e`
- | `if t then i1 else i2 (condicional)`
- | `while t do i (ciclo)`
- | `i; i` (sequência de instruções)
- | `skip` (não fazer nada)

## Testes (de validade)

$t ::=$

- | `b` (constante booleana)
- | `e == e`
- | `e > e`
- | `e < e`
- | `!t (t  $\Rightarrow$  true  $\rightarrow$  !t  $\Rightarrow$  false / t  $\Rightarrow$  false  $\rightarrow$  !t  $\Rightarrow$  true)`

As palavras **set**, **print**, **if**, **then**, **else**, **while** e **do** são reservadas pela linguagem.

Assim, podemos construir instruções como as que se seguem:

```
x = 1 + 5 - 2  
print (set y = 10 in x + y)  
if x==2 then print x else skip
```

Apresentada a sintaxe, eleva-se a necessidade de um **sistema de tipos** adequado:

## Casos base

$$\frac{}{\Gamma \vdash n : \text{int}}$$

$$\frac{}{\Gamma \vdash b : \text{boolean}}$$

$$\frac{}{\Gamma \vdash \text{var} : \Gamma(\text{var})}$$

## Operadores

$$\frac{}{\Gamma \vdash + : \text{int} * \text{int} \rightarrow \text{int}}$$

$$\frac{}{\Gamma \vdash - : \text{int} * \text{int} \rightarrow \text{int}}$$

$$\frac{}{\Gamma \vdash * : \text{int} * \text{int} \rightarrow \text{int}}$$

$$\frac{}{\Gamma \vdash / : \text{int} * \text{int} \rightarrow \text{int}}$$

## Outras regras

$$\frac{\Gamma \vdash e : T1}{\Gamma \vdash (e) : T1}$$

$$\frac{\Gamma \vdash e_1 : T1 \quad \Gamma + \text{var} : T1 \vdash e_2 : T2}{\Gamma \vdash \text{set var} = e_1 \text{ in } e_2 : T2}$$


Seguidamente, apresentamos a **semântica operacional *small-steps*** ( $E, s \rightarrow (\dots) \rightarrow E', \text{skip}$ ) da nossa linguagem Arith:

*Atribuição*

$$\frac{E, e \twoheadrightarrow v}{E, x = e \rightarrow E \{x \rightarrow v\}, \text{skip}}$$

*Impressão*

$$\frac{E, e \twoheadrightarrow v}{E, \text{print } e \rightarrow E, \text{skip}}$$

 Impressão no *stdout* de "v"

*Condiconal*

$$\frac{E, t \twoheadrightarrow \text{true}}{E, \text{if } t \text{ then } i_1 \text{ else } i_2 \rightarrow E, i_1}$$

$$\frac{E, t \rightarrow \text{false}}{E, \text{if } t \text{ then } i_1 \text{ else } i_2 \rightarrow E, i_2}$$

*Ciclo*

$$\frac{E, t \rightarrow \text{true}}{E, \text{while } t \text{ do } i \rightarrow E, i; \text{while } t \text{ do } i}$$

$$\frac{E, t \rightarrow \text{false}}{E, \text{while } t \text{ do } i \rightarrow E, \text{skip}}$$

*Sequência de instruções*

$$\frac{}{E, \text{skip}; i \rightarrow E, i}$$

$$\frac{E, i_1 \rightarrow E_1, i_1'}{E, i_1; i_2 \rightarrow E_1, i_1'; i_2}$$