

Construção de Compiladores

Tarefa 1

Semestre 1/2015

1 Objetivos e organização

Esta tarefa possui como objetivos:

- Aumentar a experiência de uso da linguagem fonte que um determinado grupo utilizará.
- Possibilitar uma compreensão mais aprofundada dos códigos Assembly da plataforma selecionada para o grupo, que será a sua linguagem alvo.
- Desenvolver a habilidade de uso das ferramentas disponíveis para a plataforma selecionada.

Os programas implementados serão apresentados por todos os membros do grupo, com 2 alunos, em sala de aula (laboratório 3) no dia *30 de março de 2015*, impreterivelmente. Nessa oportunidade cada membro será questionado e convidado a executar alguns programas escolhidos aleatoriamente.

Na data da entrega deverá ser encaminhado um relatório escrito em L^AT_EX assim como todos os códigos fontes utilizados, inclusive os arquivos `.tex` usados na elaboração do relatório.

O relatório deve ser claro e conter, entre outras informações, o nome e o propósito do algoritmo implementado, o código do programa desse algoritmo e os códigos para os testes desses programas.

A disciplina usa o Linux como sistema operacional e OCaml como linguagem de implementação.

2 Enunciado

A tarefa desta semana consiste em:

- Instalar o Ubuntu 14.04 (ou uma versão mais recente) em sua máquina.
- Instalar e utilizar a plataforma utilizada por cada grupo (JVM, Dalvik, LLVM, etc).
- Converter os programas dados na próxima seção, apresentados na forma de pseudo códigos, para a sua linguagem fonte (MiniC, MinJava, etc.)
- Escrever os equivalentes de cada programa na Assembly de seu grupo (Jasmin para JVM, Smali para Dalvik, etc).

3 Informações importantes

entrega 30/03/2015 (segunda-feira)

e-mail entregas.ufu@gmail.com

assunto Compiladores - Tarefa 1

Não esquecer de anexar o relatório, em L^AT_EX, contendo os programas, as traduções e toda a informação necessária para a instalação, compilação e execução dos programas. Colocar também as saídas de cada programa.

4 Programas

Note que o comando *escreval* escreve os seus argumentos para a tela, saltando uma linha ao final. Ao passo que *escreva* apenas escreve seus argumentos para a tela.

4.1 Nano programas

Listagem 1: Módulo mínimo que caracteriza um programa

```
algoritmo "nano01"
var
início
fim_algoritmo
```

Listagem 2: Declaração de uma variável

```
algoritmo "nano02"
var
  n : inteiro
início
fim_algoritmo
```

Listagem 3: Atribuição de um inteiro a uma variável

```
algoritmo "nano03"
var
  n : inteiro
início
  n ← 1
fim_algoritmo
```

Listagem 4: Atribuição de uma soma de inteiros a uma variável

```
algoritmo "nano04"
var
  n : inteiro
início
  n ← 1 + 2
fim_algoritmo
```

Listagem 5: Inclusão do comando de impressão

```
algoritmo "nano05"
var
  n : inteiro
início
  n ← 2
  escreva(n)
fim_algoritmo
```

Listagem 6: Atribuição de uma subtração de inteiros a uma variável

```
algoritmo "nano06"
var
  n : inteiro
início
  n ← 1 - 2
  escreva(n)
fim_algoritmo
```

Listagem 7: Inclusão do comando condicional

```
algoritmo "nano07"
var
  n : inteiro
início
  n ← 1
  se n = 1 então
    escreva(n)
  fim_se
fim_algoritmo
```

Listagem 8: Inclusão do comando condicional com parte senão

```
algoritmo "nano08"
var
  n : inteiro
início
  n ← 1
  se n = 1 então
    escreva(n)
  senão
    escreva(0)
  fim_se
fim_algoritmo
```

Listagem 9: Atribuição de duas operações aritméticas sobre inteiros a uma variável

```
algoritmo "testel_9"
var
  n : inteiro
início
  n ← 1 + 1 / 2
  se n = 1 então
```

```
    escreva(n)
senão
    escreva(0)
fim_se
fim_algoritmo
```

Listagem 10: Atribuição de duas variáveis inteiras

```
algoritmo "nano10"
var
    n, m : inteiro
início
    n ← 1
    m ← 2
    se n = m então
        escreva(n)
    senão
        escreva(0)
    fim_se
fim_algoritmo
```

Listagem 11: Introdução do comando de repetição enquanto

```
algoritmo "nano11"
var
    n, m, x : inteiro
início
    n ← 1
    m ← 2
    x ← 5
    enquanto x > n faça
        n ← n + m
        escreva(n)
    fim_enquanto
fim_algoritmo
```

Listagem 12: Comando condicional aninhado em um comando de repetição

```
algoritmo "nano12"
var
    n, m, x : inteiro
início
    n ← 1
    m ← 2
    x ← 5
    enquanto x > n faça
        se n = m então
            escreva(n)
        senão
            escreva(0)
        fim_se
        x ← x - 1
    fim_enquanto
```

fim_algoritmo

4.2 Micro programas

Listagem 13: Converte graus Celsius para Fahrenheit

```
algoritmo "micro01"
/*
  Função: Ler uma temperatura em graus Celsius e apresentá-la
          convertida em graus Fahrenheit. A fórmula de conversão é:
           $F = (9 \times C + 160) / 5$ ,
          sendo F a temperatura em Fahrenheit e C a temperatura em
          Celsius.
*/
var
cel, far: real

início

escreval("    Tabela de conversão: Celsius -> Fahrenheit")
escreva("Digite a temperatura em Celsius: ")
leia(cel)
far  $\leftarrow$  (9*cel+160)/5
escreval("A nova temperatura é: ",far," F")

fim_algoritmo
```

Listagem 14: Ler dois inteiros e decide qual é maior

```
algoritmo "micro02"
/* Função : Escrever um algoritmo que leia dois valores inteiro
            distintos e informe qual é o maior.
*/

var
num1, num2: inteiro

início

escreva("Digite o primeiro número: ")
leia(num1)
escreva("Digite o segundo número: ")
leia(num2)

se num1 > num2 então
    escreva("O primeiro número ",num1," é maior que o segundo",num2
    )
senão
    escreva("O segundo número",num2," é maior que o primeiro",num1)
fim_se

fim_algoritmo
```

Listagem 15: Lê um número e verifica se ele está entre 100 e 200

```
algoritmo "micro03"
/* Função : Faça um algoritmo que receba um número e diga se este
    número está no intervalo entre 100 e 200.
*/
var
numero: inteiro

início

escreva("Digite um número: ")
leia(numero)
se numero >= 100 então
    se numero <= 200 então
        escreval("O número está no intervalo entre 100 e 200")
    senão
        escreval("O número não está no intervalo entre 100 e 200")
    fim_se
senão
    escreval("O número não está no intervalo entre 100 e 200")
fim_se

fim_algoritmo
```

Listagem 16: Lê números e informa quais estão entre 10 e 150

```
algoritmo "micro04"
/*
    Função: Ler 5 números e ao final informar quantos número(s)
    est(á)ão no intervalo entre 10 (inclusive) e 150 (inclusive).
*/
var
x, num, intervalo: inteiro

início

para x de 1 até 5 faça
    escreva("Digite um número: ")
    leia(num)
    se num >= 10 então
        se num <= 150 então
            intervalo  $\leftarrow$  intervalo + 1
        fim_se
    fim_se
fim_para

escreval("Ao total, foram digitados ", intervalo, " números no
    intervalo entre 10 e 150")

fim_algoritmo
```

Listagem 17: Lê strings e caracteres

```
algoritmo "micro05"
/* Função : Escrever um algoritmo que leia o nome e o sexo de 56
            pessoas e informe o nome e se ela é homem ou mulher.
            No final informe o total de homens e de mulheres.
*/
var
nome, sexo: caractere
x, h, m: inteiro

início

para x de 1 até 5 faca
    escreva("Digite o nome: ")
    leia(nome)
    escreva("H - Homem ou M - Mulher: ")
    leia(sexo)
    escolha sexo
        caso "H"
            h ← h + 1
        caso "M"
            m ← m + 1
        outrocaso
            escreval("Sexo só pode ser H ou M!")
    fim_escolha
fim_para

escreval("Foram inseridos",h," Homens")
escreval("Foram inseridos",m," Mulheres")

fim_algoritmo
```

Listagem 18: Escreve um número lido por extenso

```
algoritmo "micro06"
/* Função : Faça um algoritmo que leia um número de 1 a 5 e o
            escreva por extenso. Caso o usuário digite um número que
            não esteja neste intervalo, exibir mensagem: número inválido.
*/
var
numero: inteiro

início
    escreva("Digite um número de 1 a 5: ")
    leia(numero)
    escolha numero
        caso 1
            escreval("Um")
        caso 2
            escreval("Dois")
        caso 3
            escreval("Três")
```

```

        caso 4
            escreval("Quatro")
        caso 5
            escreval("Cinco")
        outrocaso
            escreval("Número Inválido!!!")
    fim_escolha
fim_algoritmo

```

Listagem 19: Decide se os números são positivos, zeros ou negativos

```

algoritmo "micro07"
/* Função : Faça um algoritmo que receba N números e mostre
           positivo, negativo ou zero para cada número.
*/
var
    programa, numero: inteiro
    opc: caractere

início

programa ← 1
enquanto programa = 1 faça
    escreva("Digite um número: ")
    leia(numero)
    se numero > 0 então
        escreval("Positivo")
    senão
        se numero = 0 então
            escreval("O número é igual a 0")
        fim_se
        se numero < 0 então
            escreval("Negativo")
        fim_se
    fim_se

    escreva("Deseja finalizar? (S/N) ")
    leia(opc)
    se opc = "S" então
        programa ← 0
    fim_se
fim_enquanto

fim_algoritmo

```

Listagem 20: Decide se um número é maior ou menor que 10

```

algoritmo "micro08"

var
    numero: inteiro

```


início

```
numero ← 1
enquanto numero <> 0 faça
    escreva ("Digite um número: ")
    leia (numero)

    se (numero > 10) então
        escreval ("O número ", numero, " é maior que 10")
    senão
        escreval ("O número ", numero, " é menor que 10")
    fim_se
fim_enquanto
```

fim_algoritmo

Listagem 21: Cálculo de preços

algoritmo "micro09"

var

preço, venda, novo_preço: **real**

início

```
escreva ("Digite o preço: ")
leia (preço)
escreva ("Digite a venda: ")
leia (venda)

se (venda < 500) OU (preço < 30)
então novo_preço ← preço + 10/100 * preço
senão se (vendas >= 500 E venda < 1200) OU
    (preço >= 30 E preço < 80)
    então novo_preço ← preço + 15/100 * preço
senão se venda >= 1200 OU preço >= 80
    então novo_preço ← preço - 20/100 * preço
    fim_se
fim_se
fim_se

escreval ("O novo preço é ", novo_preço)
```

fim_algoritmo

Listagem 22: Calcula o fatorial de um número

algoritmo "micro10"

```
/* Função : recebe um número e calcula recursivamente o fatorial
desse número.
*/
```

var

numero: **inteiro**

```

fat: inteiro

início

escreva("Digite um número: ")
leia(numero)
fat ← fatorial(numero)

escreva("O fatorial de ")
escreva(numero)
escreva(" é ")
escreval(fat)

fim_algoritmo

função fatorial(n: inteiro) : inteiro
    se n <= 0 então
        retorne 1
    senão
        retorne n * fatorial(n-1)
fim_função fatorial

```

Listagem 23: Decide se um número é positivo, zero ou negativo com auxílio de uma função

```

algoritmo "micro11"
/* Função : recebe um número e verifica se o número é positivo, nulo
    ou negativo com auxílio de uma função.
*/

var
numero: inteiro
x: inteiro

início

escreva("Digite um número: ")
leia(numero)
x ← verifica(numero)
se x = 1
então escreval("Número positivo")
senão se x = 0
    então escreval("Zero")
    senão escreval("Número negativo")
fim_se
fim_se
fim_algoritmo

função verifica(n: inteiro) : inteiro
    var
        res: inteiro
    se n > 0 então
        res ← 1
    senão se n < 0

```

```
        então res ← -1
        senão res ← 0
    fim_se
fim_se
retorne res
fim_função verifica
```