Primeiro Trabalho Prático de AOC I
Assembly MIPS

Pablo Cecilio Oliveira
Marcilene Reis

1 Introdução

Mambojambo introduzindo sobre o que é o documento, as questões, Mars, etc.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

2 Implementação

2.1 Questão 1

O procedimento para calcular o *cosseno* de um ângulo segundo a série de Taylor abaixo:

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

A função *seno* chama funções para calcular o fatorial e a função potência de x. O usuário digita o ângulo em radianos e a quantidade de termos da série ($n > 0$).

2.2 Questão 2

O programa lê uma `string` de um arquivo denominado `entrada.txt` e gera um novo arquivo de nome `saida.txt` contendo uma nova `string` processada.

A `string` gerada no arquivo de saída possui seus caracteres maiúsculos trocados por minúsculos, e seus caracteres minúsculos trocados por maiúsculos.

Para iniciar o processamento, inicialmente os dados dos arquivos a serem carregados são declarados, assim como é definido o espaço reservado para o buffer de leitura do arquivo de entrada.

Listing 1: .data

```
.data
fin:      .asciiz "entrada.txt"
fout:     .asciiz "saida.txt"
string:   .space 1024
```

Inicia .text. A chamada para os procedimentos e a passagem das referencias.

Listing 2: main

```
main:
la  $a0, fin
jal leArquivo
la  $a0, string
jal manipulaString
la  $a0, fout
jal salvaArquivo
jal exit
```

Procedimento `leArquivo` mais comentários a respeito de seu funcionamento por etapas e breve sumario de conclusão desse.

Listing 3: leArquivo

```
leArquivo:
li   $v0, 13        # system call: Abre o arquivo
li   $a1, 0         # flag para somente leitura
li   $a2, 0         # modo ignorado
syscall             # Abra o arquivo!
move $t0, $v0       # Salva o descriptor em $t0
li   $v0, 14        # system call: Lendo o arquivo
move $a0, $t0       # Carrega o descriptor do arquivo
la   $a1, string    # endereco buffer
li   $a2, 255       # hardcoded buffer length
syscall             # Leia o arquivo!
move $s0, $v0       # Salva strlen em $s0
li   $v0, 16        # system call: Fecha o arquivo
syscall             # Feche o arquivo!
jr $ra
```

Procedimento `manipulaString` mais comentários a respeito de seu funcionamento por etapas e breve sumario de conclusão desse.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur

auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Listing 4: manipulaString

```
manipulaString:
Loop:
lb   $t2, ($a0)          # get the first byte pointed
beq  $t2, $zero, End     # if is equal to zero, End
slti $t1, $t2, 90        # if 'A-Z'
bne  $t1, $zero, Upper
slti $t1, $t2, 122       # if 'a-z'
bne  $t1, $zero, Lower
j Continue
Upper:
slti $t1, $t2, 65
bne  $t1, $zero, Continue
addi $t2, $t2, 32
sb   $t2, ($a0)          # store it in the string
j Continue
Lower:
slti $t1, $t2, 97
bne  $t1, $zero, Continue
addi $t2, $t2, -32
sb   $t2, ($a0)          # store it in the string
j Continue
Continue:                # Continue the iteration
addi $a0, $a0, 1         # Increment the address
j Loop
End:
jr $ra
```

Procedimento `salvaArquivo` mais comentários a respeito de seu funcionamento por etapas e breve sumario de conclusão desse.

Listing 5: salvaArquivo

```
salvaArquivo:
li   $v0, 13         # system call: Abre o arquivo
li   $a1, 1          # flag para escrita
li   $a2, 0          # modo ignorado
syscall              # Abra o arquivo!
move $t0, $v0        # Salva o descriptor em $t0
li   $v0, 15         # system call: Escrevendo no arquivo
move $a0, $t0        # Carrega o descriptor do arquivo
la   $a1, string     # endereco do buffer
move $a2, $s0        # carrega strlen de $s0
syscall              # Escreva no arquivo!
li   $v0, 16         # system call: Fecha o arquivo
syscall              # Feche o arquivo!
jr $ra
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Listing 6: exit

```
li $v0, 10          # system call: Saia do programa
syscall             # Saia!
```

Referências

[1] Missouri State University. Syscall functions available in mars. `http://courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html`. [Acesso em: 13-Outubro-2018].

O histórico do desenvolvimento desse trabalho se encontra online em: `https://github.com/Durfan/ufsj-aoc1-tp1`.