

**Universidade Federal de Campina Grande Unidade Acadêmica de Sistemas e
Computação Disciplina: Organização e Arquitetura de Computadores Profa. Joseana
Macêdo Fachine Régis de Araújo**

Lista de Exercícios Prática 04 (Processador - Assembly)

Alunos:

Bruno Machado de Assis - 119110401

Lucas Gabriel Soares de Almeida - 119110385

Matheus Vinicius Benevides Lima - 119210384

1.

**addi a0, zero, 4 #Definindo que vai printar string
la a1, helloworld**

ecall

nop

.data

helloworld: .asciiz "Hello, World!"

The screenshot displays a RISC-V assembly simulator interface. At the top, there are control buttons: Run (highlighted in green), Step, Prev, Reset, and Dump. Below these is a table with three columns: Machine Code, Basic Code, and Original Code. The table lists several instructions, including 'addi x10 x0 4', 'auipc x11 65536', 'la a1, helloworld', 'addi x11 x11 -4', 'ecall', and 'addi x0 x0 0'. Below the table is a text box containing the output 'Hello, World!'. To the right, there is a 'Registers' panel showing a table of registers (a0, a1, a2, a3) with their current values. The 'Memory' panel shows a table of memory addresses and their contents, including the string 'Hello, World!' stored in memory. At the bottom right, there is a 'Display Settings' dropdown menu set to 'ASCII'.

Machine Code	Basic Code	Original Code
0x00400513	addi x10 x0 4	addi a0, zero, 4 #Definindo que vai printar string
0x10000597	auipc x11 65536	la a1, helloworld
0xffc58593	addi x11 x11 -4	la a1, helloworld
0x00000073	ecall	ecall
0x00000013	addi x0 x0 0	nop

Hello, World!

Address	+0	+1	+2	+3
0x10000018				
0x10000014				
0x10000010				
0x1000000c	'!'			
0x10000008	'o'	'r'	'l'	'd'
0x10000004	'o'	','	' '	'W'
0x10000000	'H'	'e'	'l'	'l'
0xffffffffc				
0xffffffff8				
0xffffffff4				
0xffffffff0				
0xfffffdec				
0xffffffe8				

Display Settings: ASCII

2.

a)

Inicio:

a0 = 0x00000000

a1 = 0x00000000

a2 = 0x00000000

a3 = 0x00000000

1º Execução:

a0 = 0x00000005

a1 = 0x00000000

a2 = 0x00000000
a3 = 0x00000000

2º Execução:

a0 = 0x00000005
a1 = 0x0000000a
a2 = 0x00000000
a3 = 0x00000000

3º Execução:

a0 = 0x00000005
a1 = 0x0000000a
a2 = 0x00000005
a3 = 0x00000000

4º Execução:

a0 = 0x00000005
a1 = 0x0000000a
a2 = 0x00000005
a3 = 0x0000000f

5º Execução:

a0 = 0x00000005
a1 = 0x0000000a
a2 = 0x00000005
a3 = 0x0000000a

b)

Código de máquina	Instrução
0x00500513	addi a0, zero, 5
0x00a00593	addi a1, zero, 10
0x00500613	addi a2, zero, 5
0x00b506b3	add a3, a0, a1
0x40c686b3	sub a3, a3, a2

c) addi x10 x0 5

3.

a) Calcula o logaritmo de 128 na base 2.

Fazendo sucessivos deslocamentos de 1(s0) até que s0 seja igual a t0, e enquanto isso soma 1 a s1 sempre que há o deslocamento,
Ou seja, o resultado fica em s1.

Obs: sabendo que deslocar para esquerda é o mesmo que multiplicar pela base, ou seja, nesse caso, por 2.

b)

s0 = 128

s1 = 7

4.

```
lw t0, g
lw t1, h
lw t2, i
lw t3, j
```

```
add s4, t0, t1
sub s5, t2, t3
add s6, s4, s5
```

```
.data
g: .word 3
h: .word 2
i: .word 5
j: .word 1
```

t0, t1, t2, t3 são respectivamente os valores g,h,i,j

```
s4 = g+h
s5 = i-j
s6 = f
```

Registradores:

```
t0 = 0x00000003
t1 = 0x00000002
t2 = 0x00000005
t3 = 0x00000001
```

```
s4 = 0x00000005
s5 = 0x00000004
s6 = 0x00000009
```

Logo, o resultado da operação é 9.