

Lab 2
22/07/2024

1. Identificação dos participantes

Nome	RGA	Turma
Anthony Muniz Prado de Oliveira	202011722003	CC

2. Introdução.

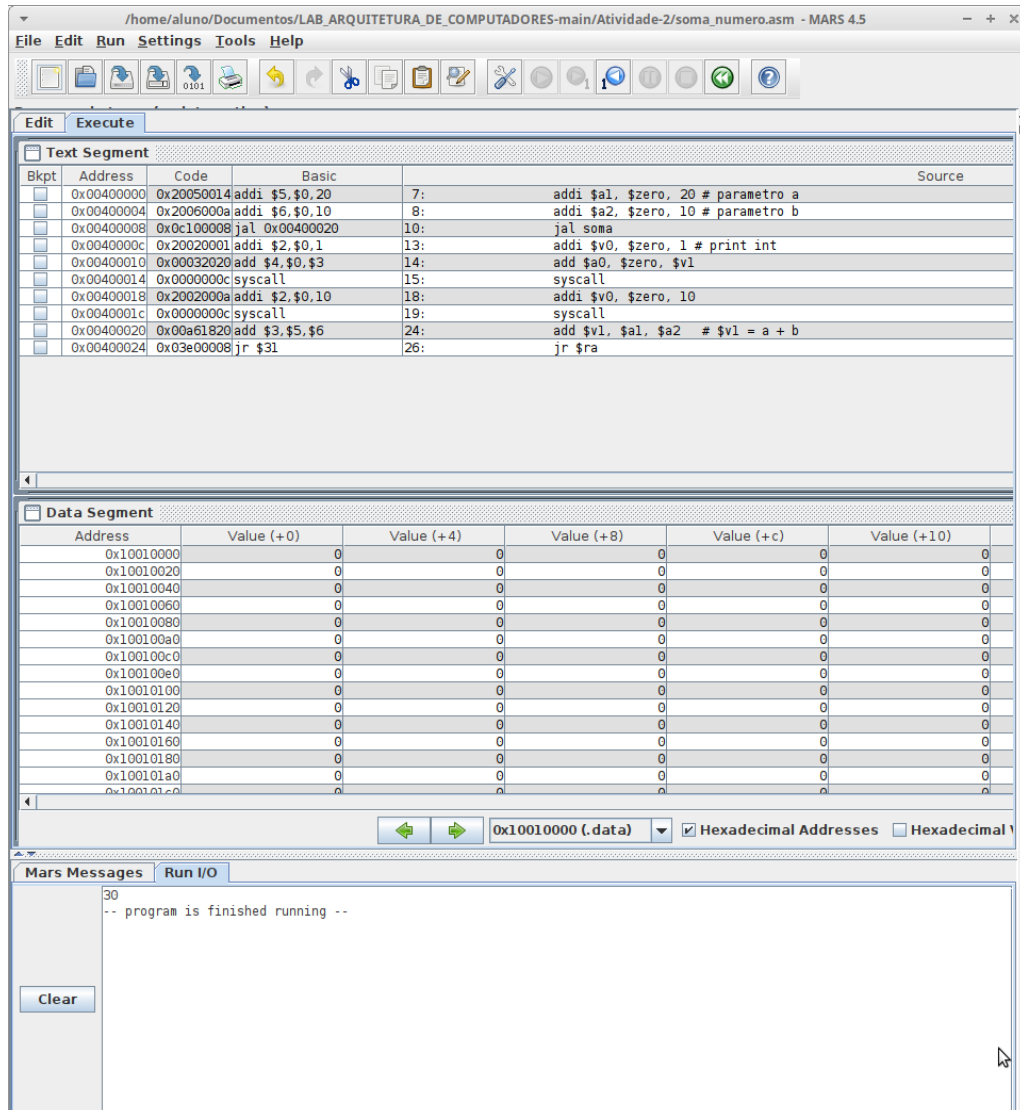
Neste trabalho vocês exercitarão seus conhecimentos sobre a ISA (*Instruction Set Architecture*) do processador MIPS de 32 bits. Para isso, vocês construirão programas em Assembly que explorarão a implementação de chamadas de funções. As implementações devem ser executadas no simulador MARS, a fim de verificar a corretude.

Para cada exercício, anexe o código fonte em um arquivo com o nome indicado, e cole um *print* da tela com o resultado final dado pelo simulador. O trabalho pode ser feito individualmente ou em dupla. Para o caso de duplas, apenas um dos alunos deve submeter o relatório no site da disciplina.

3. Exercícios.

Para cada exercício abaixo, salve o programa no arquivo indicado e cole o print da última tela do simulador.

1. Implemente uma função que receba como argumento dois números inteiros e retorne o resultado da soma desses números. Faça um programa que chame a função e apresente o resultado. Salve no arquivo: soma_numero.asm.



2. Faça uma função recursiva que receba um número inteiro n e some de 1 até n. Salve no arquivo: soma_recursiva.asm.

Interface MARS 4.5 showing the assembly code for a recursive function to calculate the sum of integers from 1 to n.

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20050007	addi \$5,\$0,7	9: addi \$a1, \$zero, 7 # parametro n
	0x00400004	0x3c011001	lui \$1,4097	10: sw \$a1, origin
	0x00400008	0xac250004	sw \$5,4(\$1)	
	0x0040000c	0xc0100009	jal 0x00400024	12: jal soma_recursiva
	0x00400010	0x20020001	addi \$2,\$0,1	15: addi \$v0, \$zero, 1 # print int em v1
	0x00400014	0x0032020	add \$4,\$0,\$3	16: add \$a0, \$zero, \$v1
	0x00400018	0x0000000c	syscall	17: syscall
	0x0040001c	0x2002000a	addi \$2,\$0,10	20: addi \$v0, \$zero, 10
	0x00400020	0x0000000c	syscall	21: syscall
	0x00400024	0x18a00005	blez \$5,5	26: blez \$a1, caso_base # se a1 <= 0 va para recursao
	0x00400028	0x23bdf8	addi \$29,\$29,-8	28: addi \$sp, \$sp, -8
	0x0040002c	0xafbf0000	sw \$31,0(\$29)	29: sw \$ra, 0(\$sp) # salva retorno atual
	0x00400030	0xaf500004	sw \$5,4(\$29)	30: sw \$a1, 4(\$sp) # salva a1 atual
	0x00400034	0x20a5ffff	addi \$5,\$5,-1	33: addi \$a1, \$a1, -1 # decrementa a1
	0x00400038	0xc0100009	jal 0x00400024	35: jal soma_recursiva
	0x0040003c	0x8fa90004	lw \$9,4(\$29)	39: lw \$t1, 4(\$sp) # carrega variavel contadora
	0x00400040	0x8fbf0000	lw \$31,0(\$29)	40: lw \$ra, 0(\$sp) # carrega variavel de retorno
	0x00400044	0x3c011001	lui \$1,4097	41: lw \$t2, result # Variavel que recebe o resultado
	0x00400048	0x8c2a0000	lw \$10,0(\$1)	
	0x0040004c	0x3c011001	lui \$1,4097	42: lw \$t3, origin # Carrega variavel de comparacao (variavel n in
	0x00400050	0x8c2b0004	lw \$11,4(\$1)	
	0x00400054	0x23bd0008	addi \$29,\$29,8	44: addi \$sp, \$sp, 8
	0x00400058	0x01495020	add \$10,\$10,\$9	47: add \$t2, \$t2, \$t1
	0x0040005c	0x3c011001	lui \$1,4097	49: sw \$t2, result # salva novo valor de resultado
	0x00400060	0xac2a0000	sw \$10,0(\$1)	
	0x00400064	0x112b0001	beq \$9,\$11,1	51: beq \$t1, \$t3, end_recursivo
	0x00400068	0xc0100009	jal 0x00400024	53: jal soma_recursiva # chamada recursiva
	0x0040006c	0x3c011001	lui \$1,4097	56: lw \$v1, result # Carrega resultado final em \$t1
	0x00400070	0x8c230000	lw \$3,0(\$1)	
	0x00400074	0x03e00008	jr \$31	57: jr \$ra

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	28	7	0	0	0
0x10010020	0	0	0	0	0
0x10010040	0	0	0	0	0
0x10010060	0	0	0	0	0
0x10010080	0	0	0	0	0
0x100100a0	0	0	0	0	0
0x100100c0	0	0	0	0	0
0x100100e0	0	0	0	0	0
0x10010100	0	0	0	0	0

Mars Messages: 28 -- program is finished running --