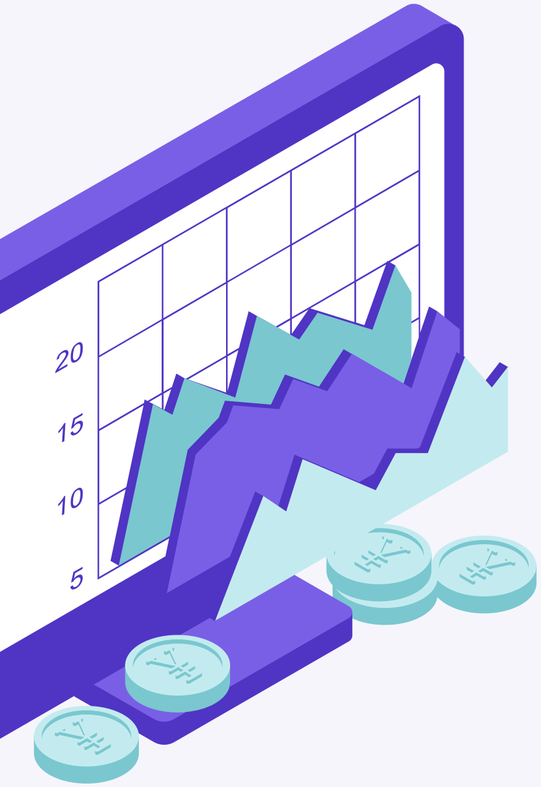
ANA BEATRIZ RODRIGUES CHAGAS - N° USP: 12734130
ANDRÉ PALACIO BRAGA TIVO - N° USP: 13835534

PROBLEMA 9: SOMA DE BITS

Exercício Programa de Organização e Arquitetura de Computadores I, 2023.1

Problema

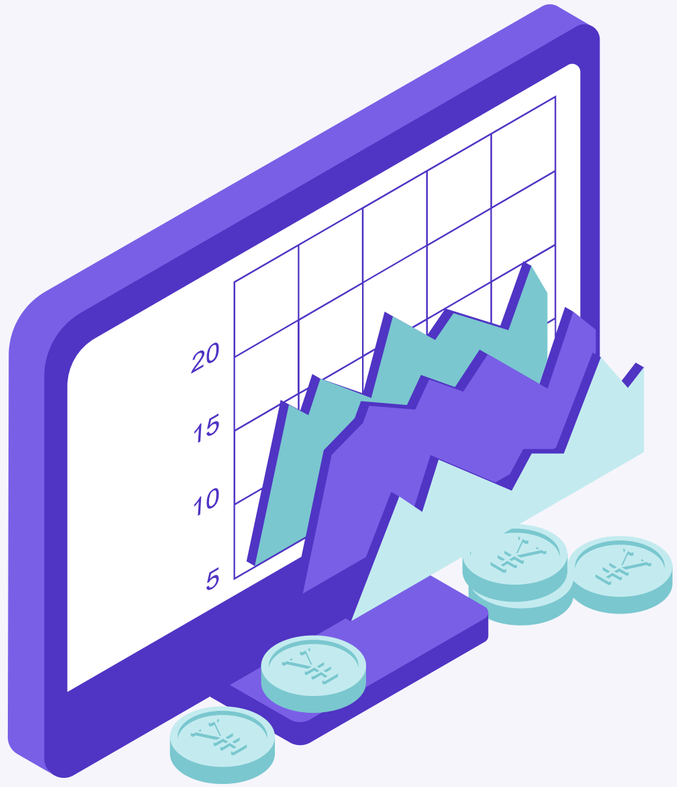


- Escreva uma função com protótipo
    - void somabit (int b1, int b2, int *vaium, int *soma);
- A função recebe três bits (inteiros entre 0 e 1) b1, b2 e *vaium e devolve um bit soma representando a soma dos três e o novo um bit "vai-um" em *vaium.

Solução

- A soma é feita através de uma comparação lógica:
  - (b1 XOR b2) XOR *vaiUm

- O *vaiUm é feito por meio de:
  - (b1 AND b2) OR ((b1 XOR b2) AND *vaiUm)

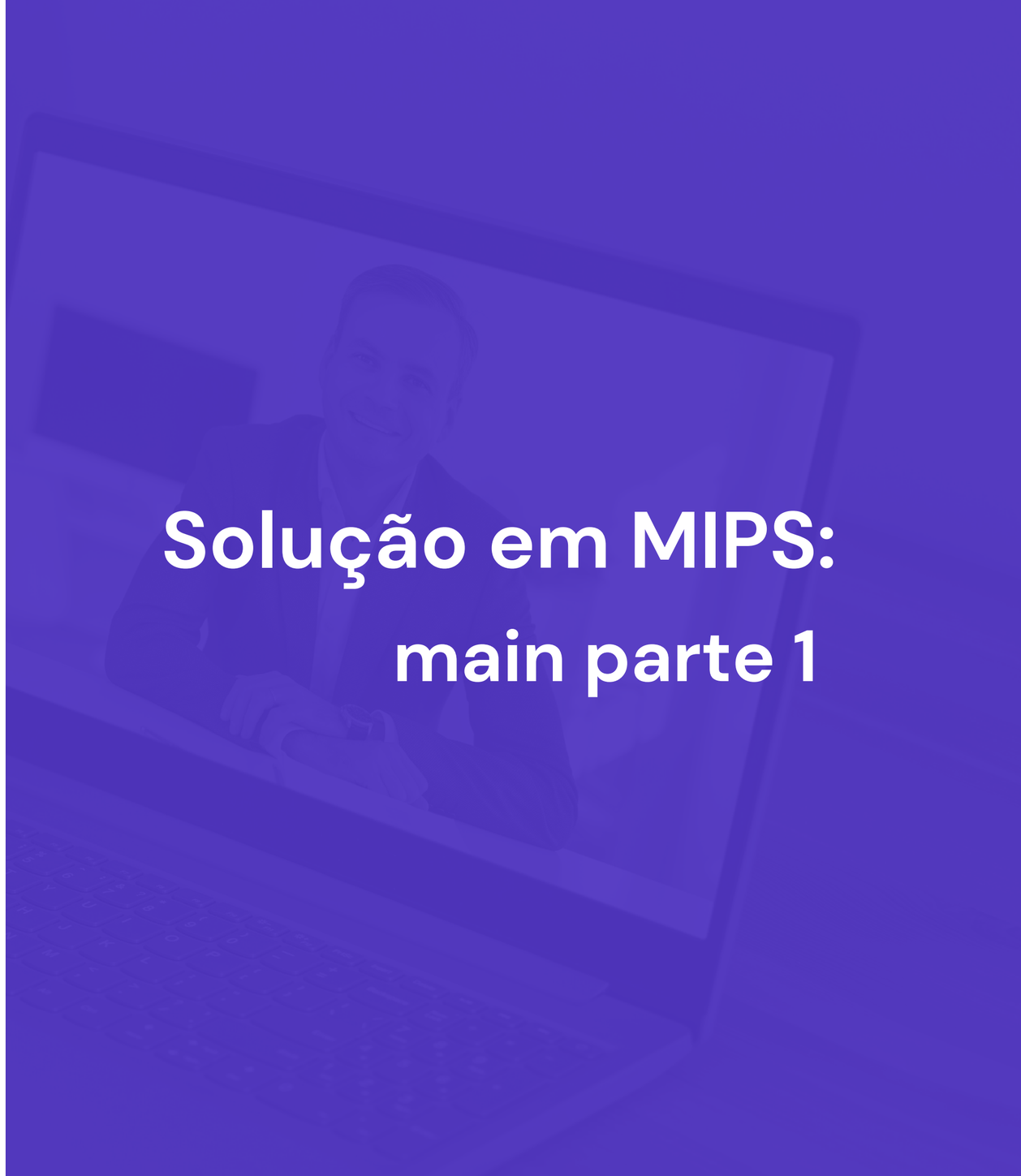


Solução em MIPS:

.data

```
.data

        # Os valores encontrados no .data NÃO devem ser alterados

        # Foi utilizada uma estrutura de repetição para considerar todos
        # os casos possíveis de b1 e b2, que são:

        # b1 = 0, b2 = 0;
        # b1 = 1 e b2 = 0 (o inverso segue a mesma retórica)
        # b1 = 1 e b2 = 1

        # O valor de vaiUm não é mudado (soma de BITS não se espera vaiUm = 1)

        # Durante o programa é utilizada a pilha (stackpointer)
        # para atualizar os valores de vaiUm e soma (os retorna a zero)

        # O resultado da soma dos bits é impresso durante o programa
        # Assim como o valor de vaiUm e o valor dos Bits da soma
        # A saída esperada é composta por:


        # O valor binário a ser somado é: b1 + b2
        # O resultado da soma é: valorDaSoma
        # O resultado vai um é: valorDoVaiUm

         b1: .byte 0 #byte 1
         b2: .byte 0 #byte 2
         vaiUm: .byte 0 # vai um da soma
         soma: .byte 0 # resultado da soma
         resSoma: .asciiz "\nO resultado da soma é: "
         resVaiUm: .asciiz "\nO resultado vai um é: "
         valorSer: .asciiz "\nO valor binário a ser somado é: "
```

```
.text
.globl main

	main:
	#TODO: fazer um for que altera o valor das variáveis

	lb, $a0, b1 # carrega b1
	lb, $a1, b2 # carrega b2
	move $s3, $a0 # salva b1 para ser comparado e alterado dentro do loop
	move $s4, $a1 # salva b2 para ser comparado e alterado dentro do loop
	# ambos em $a pois serão passados como argumentos

	loop:
	# começo da rotina de impressão do loop
	la $a0, valorSer
	li $v0, 4
	syscall

	move $a0, $s3
	li $v0, 1
	syscall

	la $a0, espacoSoma
	li $v0, 4
	syscall

	move $a0, $s4
	li $v0, 1
	syscall
	# fim da rotina de impressão do loop
	lb $t2, soma # pega a soma do .data
	lb $t3, vaiUm # pega o vaium do .data
	addiu $sp, $sp, -8 # aloca duas palavras no stack
	sw $t2, 4($sp) # salva a soma
	sw $t3, 0($sp) # salva o vai um

	move $a0, $s3 # carrega os valores, atualizados pelo loop, de b1 e b2
```



Solução em MIPS:

main parte 1

```
move $a1, $s4

jal somabit # mesma coisa que passar em C

# somabit(b1, b2, &soma, &vaium)
# como está sendo pedido no enunciado
# como é uma função void, registradores de resultado não são usados

lw $t0, 4($sp) # soma
lw $t1, 0($sp) # vaium
# pega os valores salvos no stack
# a seguinte rotina imprime os valores da Soma e do VaiUm
la, $a0, resSoma
li, $v0, 4
syscall

move $a0, $t0
li $v0, 1
syscall

la $a0, resVaiUm
li $v0, 4
syscall

move $a0, $t1
li $v0, 1
syscall

beq $s3, $zero, b1zero # altera  b1 para o valor 1
beq $s4, $zero, b2zero # altera b2 para o valor 1
beq $s3, $s4, continue # ao chegar nesse ponto do código, b1 = b2 = 1
j loop
continue:

addiu $sp, $sp, +8 # encerra/desaloca o espaço no stackpointer

end:
li $v0, 10 # rotina que encerra o programa
syscall

b1zero:
li $s3, 1 # altera o valor de b1 e reinicia o loop
j loop
```

Solução em MIPS: main parte 2

```
b2zero:
li $s4, 1 # altera o valor de b2 e reinicia o loop
j loop
somabit:

# $s0 SOMA
# $s1 VAIUM
move $t1, $a0 # salva os argumentos nos registradores temporários
move $t2, $a1

lw $s0, 4($sp) #soma
lw $s1, 0($sp) #vaium
# ^ pega os valores de soma e vaium do stackpointer

xor $t0, $t1, $t2 # (b1 ^b2) = t0
xor $t0, $t0, $s0 # t0 ^ vaium
move $s0, $t0 # salva o valor em s0

and $t0, $t1, $t2 # (b1 & b2) = $t0
xor $t3, $t1, $t2 # (b1 ^ b2) = $t3
and $t3, $t3, $s1 # ($t3 & vaium) = $t3
or $s1, $t0, $t3 # ($t0 | $t3)

# salva os valores atualizados de soma e vaium no stack
sw $s0 4($sp) # salva a soma
sw $s1, 0($sp) # salva o vai um


jr $ra
```

Solução em MIPS:

main parte 3

SAÍDA

```
O valor binário a ser somado é: 0 + 0
O resultado da soma é: 0
O resultado vai um é: 0
O valor binário a ser somado é: 1 + 0
O resultado da soma é: 1
O resultado vai um é: 0
O valor binário a ser somado é: 1 + 1
O resultado da soma é: 0
O resultado vai um é: 1
```