Exercício 04

Nome do aluno:

Digite seu nome aqui

Objetivo

Consolidar o uso da instrução de desvio condicional em RISC-V

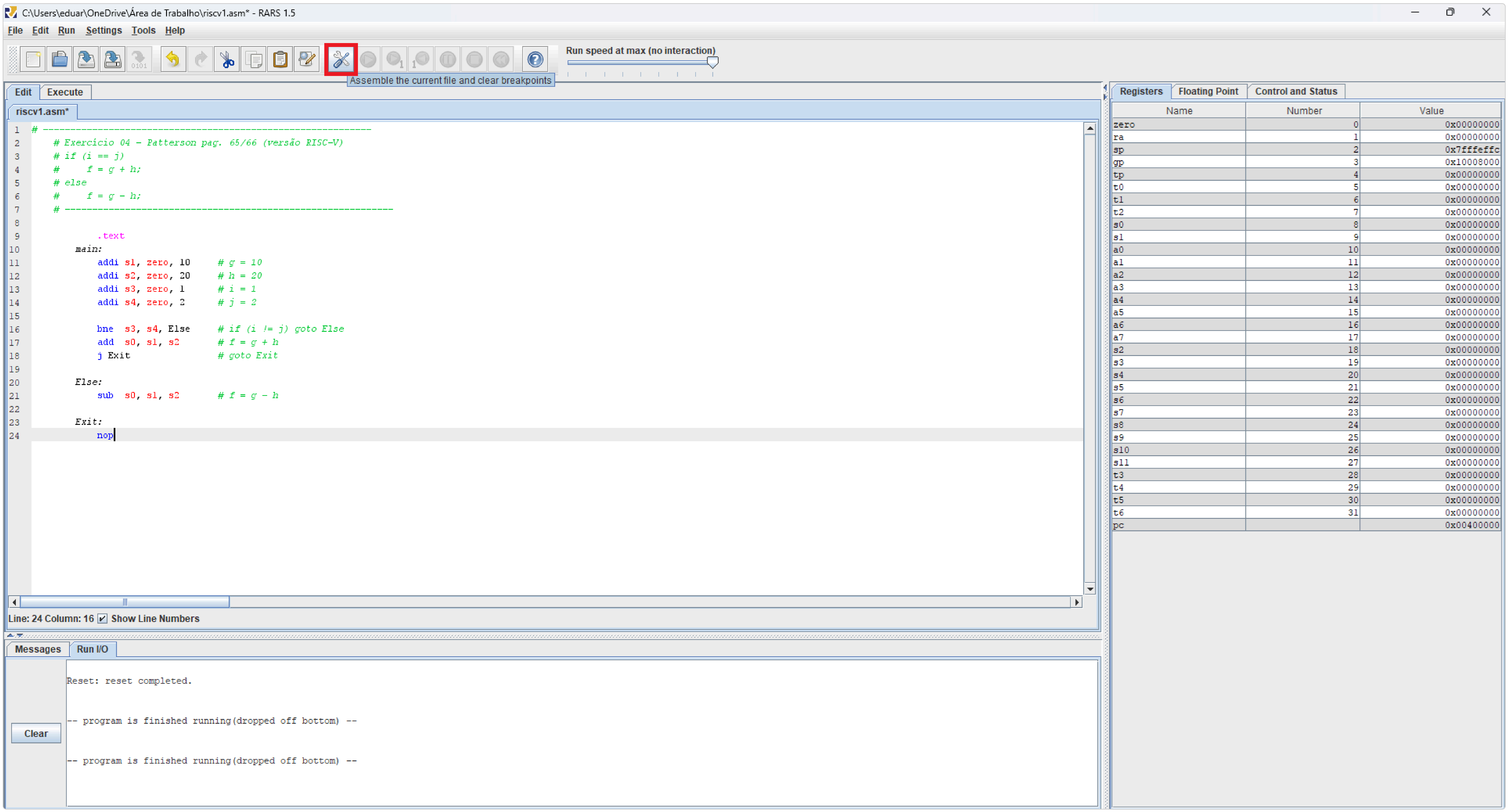
Instruções

1. Abra o simulador de linguagem RISC-V.
2. No editor de texto do simulador, transcreva o código abaixo:

```
# ------------------------------------------------------------
# Exercício 04 - Patterson pag. 65/66 (versão RISC-V)
# if (i == j)
#     f = g + h;
# else
#     f = g - h;
# ------------------------------------------------------------

        .text
    main:
        addi s1, zero, 10     # g = 10
        addi s2, zero, 20     # h = 20
        addi s3, zero, 1      # i = 1
        addi s4, zero, 2      # j = 2

        bne  s3, s4, Else     # if (i != j) goto Else
        add  s0, s1, s2       # f = g + h
        j Exit                # goto Exit

    Else:
        sub  s0, s1, s2       # f = g - h

    Exit:
        nop
```

Montagem e Execução

Clique no botão **Assemble** para montar o programa.


C:\Users\eduar\OneDrive\Área de Trabalho\riscv1.asm* - RARS 1.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Assemble the current file and clear breakpoints
Edit Execute
riscv1.asm*
Registers Floating Point Control and Status
Line: 24 Column: 16 Show Line Numbers
Messages Run I/O
Clear

```
# -------------------------------------------------------------
    # Exercício 04 - Patterson pag. 65/66 (versão RISC-V)
    # if (i == j)
    #     f = g + h;
    # else
    #     f = g - h;
    # -------------------------------------------------------------

            .text
        main:
            addi s1, zero, 10     # g = 10
            addi s2, zero, 20     # h = 20
            addi s3, zero, 1      # i = 1
            addi s4, zero, 2      # j = 2

            bne  s3, s4, Else     # if (i != j) goto Else
            add  s0, s1, s2       # f = g + h
            j Exit                # goto Exit

        Else:
            sub  s0, s1, s2       # f = g - h

        Exit:
            nop
```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400000

```
Reset: reset completed.


-- program is finished running (dropped off bottom) --


-- program is finished running (dropped off bottom) --
```

Observe que fazendo uso da instrução **addi** é possível atribuir valores aos registradores **s1**, **s2**, **s3** e **s4**, conforme segue:

Registrador	Número do Registrador	Valor
s1	9	10
s2	18	20
s3	19	1
s4	20	2

Obs:
O endereço **1** não se refere ao registrador **s1**, mas sim ao registrador **ra**.
O endereço do registrador **s1** é **9** (0x9). Ele pode ser referenciado digitando-se **x9**, bem como **s1**.

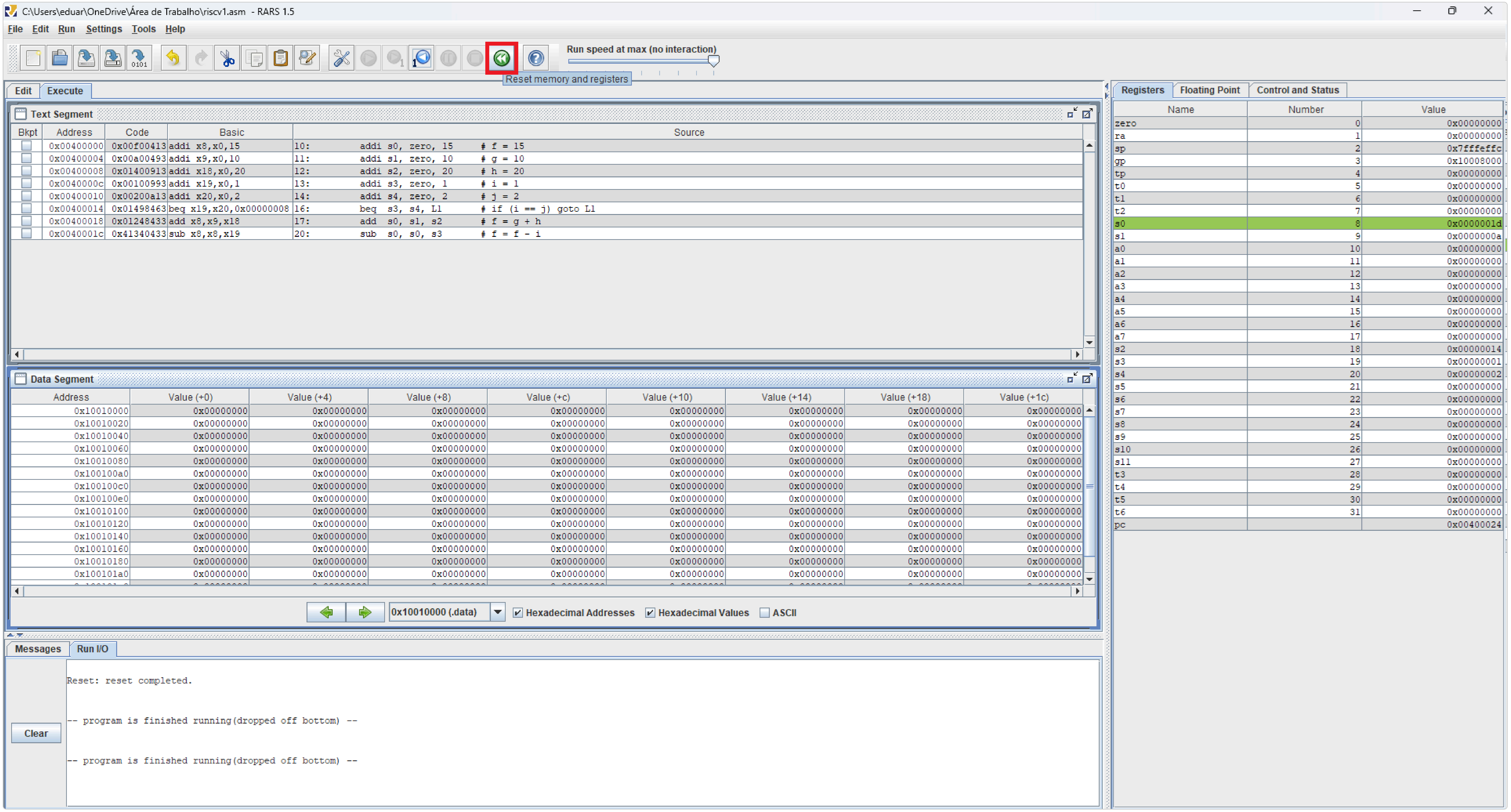
Faça a execução passo-a-passo do programa e, a cada instrução, preencha a tabela abaixo cada vez que o valor de um registrador ou posição da memória de dados for modificado.

Antes da execução da instrução		**Depois da execução da instrução**				
PC	**Instrução**	**R8**	**R9**	**R18**	**R19**	**R20**
		(s0)	**(s1)**	**(s2)**	**(s3)**	**(s4)**
0x00400000	**addi s1, zero, 10**		**0x0000000A**			
`0x00400004`	`addi s2, zero, 20`		`0x0000000A`	`0x00000014`		
`0x00400008`	`addi s3, zero, 1`		`0x0000000A`	`0x00000014`	`0x00000001`	
`0x0040000c`	`addi s4, zero, 2`		`0x0000000A`	`0x00000014`	`0x00000001`	`0x00000002`
`0x00400010`	`bne s3, s4, Else`		`0x0000000A`	`0x00000014`	`0x00000001`	`0x00000002`
`0x0040001c`	`sub s0, s1, s2`	`0xfffffff6`	`0x0000000A`	`0x00000014`	`0x00000001`	`0x00000002`
`0x00400020`	`nop`					

Altere as instruções **addi** para atribuir os seguintes valores aos registradores **s1**, **s2**, **s3**, e **s4 (note que agora s3 e s4 são iguais a 1)**.

Registrador	**Número do Registrador**	**Valor**
s1	9	10
s2	18	20
s3	19	1
s4	20	1

Recarregue o programa com a opção **Reset**.


C:\Users\eduar\OneDrive\Área de Trabalho\riscv1.asm - RARS 1.5
File
Edit
Run
Settings
Tools
Help
Run speed at max (no interaction)
Reset memory and registers
Edit
Execute
Messages
Run I/O
Clear
0x10010000 (.data)
Registers
Floating Point
Control and Status

Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x00f00413	addi x8,x0,15	10: addi s0, zero, 15 # f = 15
☐	0x00400004	0x00a00493	addi x9,x0,10	11: addi s1, zero, 10 # g = 10
☐	0x00400008	0x01400913	addi x18,x0,20	12: addi s2, zero, 20 # h = 20
☐	0x0040000c	0x00100993	addi x19,x0,1	13: addi s3, zero, 1 # i = 1
☐	0x00400010	0x00200a13	addi x20,x0,2	14: addi s4, zero, 2 # j = 2
☐	0x00400014	0x01498463	beq x19,x20,0x00000008	16: beq s3, s4, L1 # if (i == j) goto L1
☐	0x00400018	0x01248433	add x8,x9,x18	17: add s0, s1, s2 # f = g + h
☐	0x0040001c	0x41340433	sub x8,x8,x19	20: sub s0, s0, s3 # f = f - i

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

☑ Hexadecimal Addresses ☑ Hexadecimal Values ☐ ASCII

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x0000001d
s1	9	0x0000000a
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000014
s3	19	0x00000001
s4	20	0x00000002
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400024

```
Reset: reset completed.

-- program is finished running(dropped off bottom) --

-- program is finished running(dropped off bottom) --
```

Faça a execução passo-a-passo do programa e, a cada instrução, preencha a tabela abaixo cada vez que o valor de um registrador ou posição da memória de dados for modificado.

Antes da execução da instrução		**Depois da execução da instrução**				
PC	**Instrução**	**R8**	**R9**	**R18**	**R19**	**R20**
		(s0)	**(s1)**	**(s2)**	**(s3)**	**(s4)**
0x00400000	**addi s1, zero, 10**		**0x0000000A**			
0x00400004	addi s2, zero, 20		0x0000000A	0x00000014		
0x00400008	addi s3, zero, 1		0x0000000A	0x00000014	0x00000001	
0x0040000C	addi s4, zero, 1		0x0000000A	0x00000014	0x00000001	0x00000001

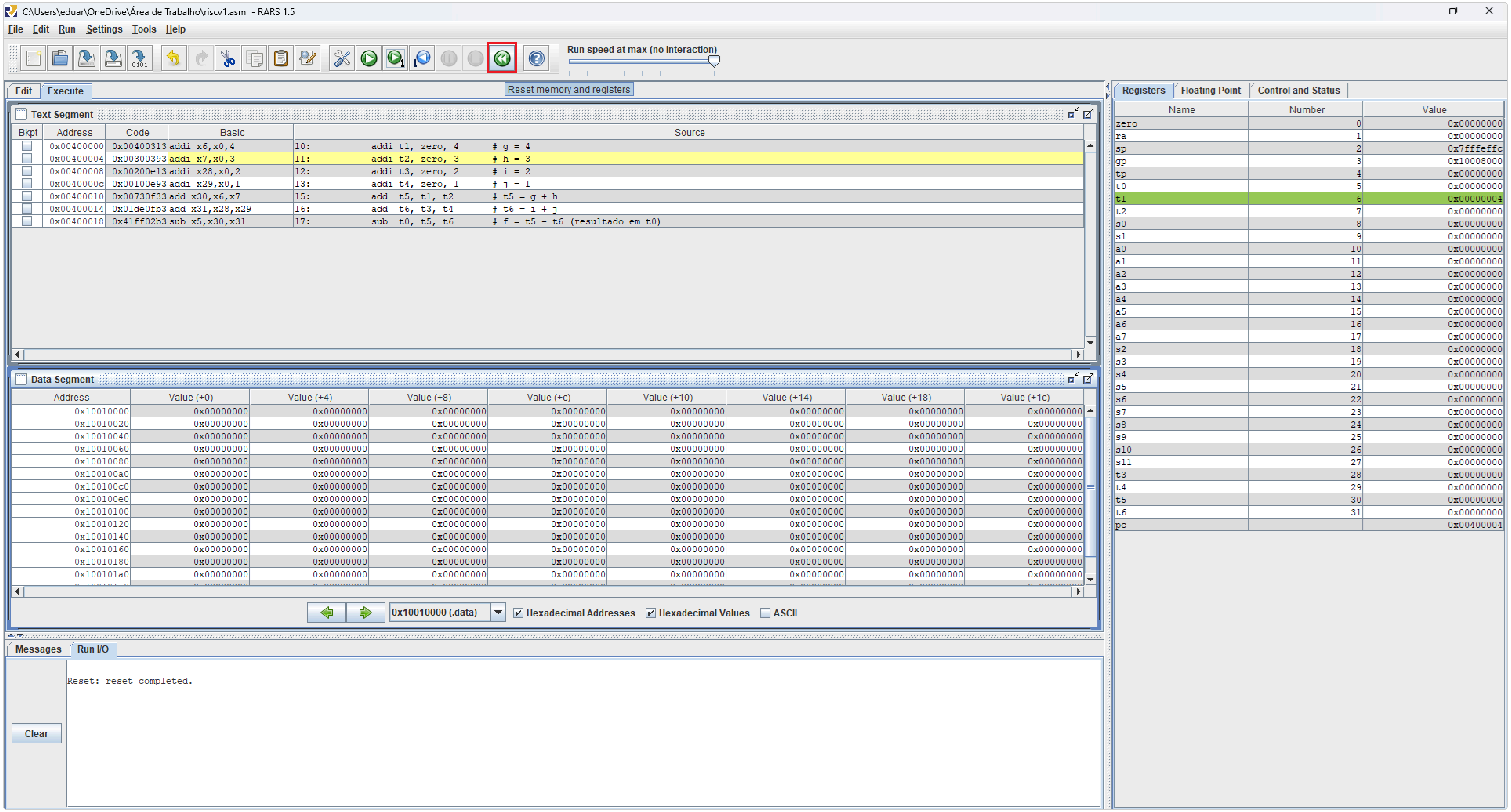
Antes da execução da instrução		**Depois da execução da instrução**				
PC	**Instrução**	**R8**	**R9**	**R18**	**R19**	**R20**
		(s0)	**(s1)**	**(s2)**	**(s3)**	**(s4)**
`0x00400010`	`bne s3, s4, Else`		`0x0000000A`	`0x00000014`	`0x00000001`	`0x00000001`
`0x00400014`	`add s0, s1, s2`	`0x0000001E`	`0x0000000A`	`0x00000014`	`0x00000001`	`0x00000001`
`0x00400018`	`nop`					

OBS: Salve o PDF em formato A2 e Paisagem para garantir que todas as informações da página fiquem visíveis

Salvar como PDF

Compare as duas tabelas e analise a diferença entre o fluxo de instruções executadas (veja a sequência de valores do **PC**) e o valor final de **s0**.

Se desejar reiniciar o programa, clique no botão **Reset**.


C:\Users\eduar\OneDrive\Área de Trabalho\riscv1.asm - RARS 1.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Reset memory and registers

Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x00400313	addi x6,x0,4	10: addi t1, zero, 4 # g = 4
☐	0x00400004	0x00300393	addi x7,x0,3	11: addi t2, zero, 3 # h = 3
☐	0x00400008	0x00200e13	addi x28,x0,2	12: addi t3, zero, 2 # i = 2
☐	0x0040000c	0x00100e93	addi x29,x0,1	13: addi t4, zero, 1 # j = 1
☐	0x00400010	0x00730f33	add x30,x6,x7	15: add t5, t1, t2 # t5 = g + h
☐	0x00400014	0x01de0fb3	add x31,x28,x29	16: add t6, t3, t4 # t6 = i + j
☐	0x00400018	0x41ff02b3	sub x5,x30,x31	17: sub t0, t5, t6 # f = t5 - t6 (resultado em t0)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) [✓] Hexadecimal Addresses [✓] Hexadecimal Values ☐ ASCII

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000004
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400004

Messages | Run I/O

```
Reset: reset completed.
```

Clear

← Voltar ao tutorial