

LABORATÓRIO DE ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

Roteiro 6

Tarso Jabbes Lima de Oliveira

PROBLEMA 1

- a) A operação realizada é a de um loop que para quando o registrador a0 atingir o valor de a2 (5). Caso contrário, o valor de a1 é shiftado um 1 bit para a esquerda e o valor de a0 é incrementado em 1 unidade.
- b) O valor de a0 é 5, de a1 é 32 e de a2 é 5.

Run Step Prev Reset Dump

Machine Code	Basic Code	Original Code
0x00100513	addi x10 x0 1	li a0, 1 # a0 = 1
0x00a505b3	add x11 x10 x10	add a1, a0, a0 # a1 = 2 * a0
0x00500613	addi x12 x0 5	li a2, 5 # a2 = 5
0x00a60863	beq x12 x10 16	loop: beq a2, a0, fim # Para a execução quando a0 >= 5
0x00159593	slli x11 x11 1	slli a1, a1, 1 # Realiza o shift para a esquerda de 1 bit de a1
0x00150513	addi x10 x10 1	addi a0, a0 1 # Incrementa o valor de a0 em 1 unidade
0xff5ff06f	jal x0 -12	j loop # Retorna para o loop quando a0 < 5
0x00000013	addi x0 x0 0	fim: nop

console output

Registers Memory

zero	0x00000000
ra (x1)	0x00000000
sp (x2)	0x7fffffff0
gp (x3)	0x10000000
tp (x4)	0x00000000
t0 (x5)	0x00000000
t1 (x6)	0x00000000
t2 (x7)	0x00000000
s0 (x8)	0x00000000
s1 (x9)	0x00000000
a0 (x10)	0x00000005
a1 (x11)	0x00000020
a2 (x12)	0x00000005
a3 (x13)	0x00000000

- c) Na instrução “j loop” o valor é -12 e na instrução “beq 02, 02, fim” é 16.

Run Step Prev Reset Dump

Machine Code	Basic Code	Original Code
0x00100513	addi x10 x0 1	li a0, 1 # a0 = 1
0x00a505b3	add x11 x10 x10	add a1, a0, a0 # a1 = 2 * a0
0x00500613	addi x12 x0 5	li a2, 5 # a2 = 5
0x00a60863	beq x12 x10 16	loop: beq a2, a0, fim # Para a execução quando a0 >= 5
0x00159593	slli x11 x11 1	slli a1, a1, 1 # Realiza o shift para a esquerda de 1 bit de a1
0x00150513	addi x10 x10 1	addi a0, a0 1 # Incrementa o valor de a0 em 1 unidade
0xff5ff06f	jal x0 -12	j loop # Retorna para o loop quando a0 < 5
0x00000013	addi x0 x0 0	fim: nop

- d) Os prints se encontram abaixo:

Print do código no editor:

```
1      li a0, 1          # a0 = 1
2      add a1, a0, a0     # a1 = 2 * a0
3      li a2, 5          # a2 = 5
4
5 loop: beq a0, a2, fim    # Para a execução quando a0 >= 5
6      slli a1, a1, 1     # Realiza o shift para a esquerda de 1 bit no valor de a1
7      addi a0, a0, 1     # Incrementa o valor de a0 em 1 unidade
8      j loop            # Retorna para o loop quando a condição de parada não é atingida
9
10 fim:  nop             # Para a execução
```

Print das instruções na aba do simulador:

Machine Code	Basic Code	Original Code
0x00100513	addi x10 x0 1	li a0, 1 # a0 = 1
0x00a505b3	add x11 x10 x10	add a1, a0, a0 # a1 = 2 * a0
0x00500613	addi x12 x0 5	li a2, 5 # a2 = 5
0x00c50863	beq x10 x12 16	loop: beq a0, a2, fim # Para a execução quando a0 >= 5
0x00159593	slli x11 x11 1	slli a1, a1, 1 # Realiza o shift para a esquerda de 1 bit no valor de a1
0x00150513	addi x10 x10 1	addi a0, a0, 1 # Incrementa o valor de a0 em 1 unidade
0xff5ff06f	jal x0 -12	j loop # Retorna para o loop quando a condição de parada não é atingida
0x00000013	addi x0 x0 0	fim: nop # Para a execução

Antes de iniciar a execução os valores na aba dos registradores e de memória são:

Registers

Memory

zero

0

ra (x1)

0

sp (x2)

2147483632

gp (x3)

268435456

tp (x4)

0

t0 (x5)

0

t1 (x6)

0

t2 (x7)

0

s0 (x8)

0

s1 (x9)

0

a0 (x10)

0

a1 (x11)

0

a2 (x12)

0

Registers

Memory

Address	+0	+1	+2	+3
0x00000018	111	-16	95	-1
0x00000014	19	5	21	0
0x00000010	-109	-107	21	0
0x0000000c	99	8	-59	0
0x00000008	19	6	80	0
0x00000004	-77	5	-91	0
0x00000000	19	5	16	0
-----	--	--	--	--
-----	--	--	--	--
-----	--	--	--	--
-----	--	--	--	--
-----	--	--	--	--
-----	--	--	--	--

Jump to

-- choose --

Up

Down

Na execução da primeira linha de código:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00100513	addi x10 x0 1	li a0, 1 # a0 = 1
0x00a505b3	add x11 x10 x10	add a1, a0, a0 # a1 = 2 * a0
0x00500613	addi x12 x0 5	li a2, 5 # a2 = 5
0x00c50863	beq x10 x12 16	loop: beq a0, a2, fim # Para a execução quando a0 >= 5
0x00159593	slli x11 x11 1	slli a1, a1, 1 # Realiza o shift para a esquerda de 1 bit no valor de a1
0x00150513	addi x10 x10 1	addi a0, a0, 1 # Incrementa o valor de a0 em 1 unidade
0xff5ff06f	jal x0 -12	j loop # Retorna para o loop quando a condição de parada não é atingida
0x00000013	addi x0 x0 0	fim: nop # Para a execução

console output

Na execução da segunda linha:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00100513	addi x10 x0 1	li a0, 1 # a0 = 1
0x00a505b3	add x11 x10 x10	add a1, a0, a0 # a1 = 2 * a0
0x00500613	addi x12 x0 5	li a2, 5 # a2 = 5
0x00c50863	beq x10 x12 16	loop: beq a0, a2, fim # Para a execução quando a0 >= 5
0x00159593	slli x11 x11 1	slli a1, a1, 1 # Realiza o shift para a esquerda de 1 bit no valor de a1
0x00150513	addi x10 x10 1	addi a0, a0, 1 # Incrementa o valor de a0 em 1 unidade
0xff5ff06f	jal x0 -12	j loop # Retorna para o loop quando a condição de parada não é atingida
0x00000013	addi x0 x0 0	fim: nop # Para a execução

console output

RegistersMemory

zero	0
ra (x1)	0
sp (x2)	2147483632
gp (x3)	268435456
tp (x4)	0
t0 (x5)	0
t1 (x6)	0
t2 (x7)	0
s0 (x8)	0
s1 (x9)	0
a0 (x10)	1
a1 (x11)	0
a2 (x12)	0
a3 (x13)	0

Na execução da terceira linha de código:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00100513	addi x10 x0 1	li a0, 1 # a0 = 1
0x00a505b3	add x11 x10 x10	add a1, a0, a0 # a1 = 2 * a0
0x00500613	addi x12 x0 5	li a2, 5 # a2 = 5
0x00c50863	beq x10 x12 16	loop: beq a0, a2, fim # Para a execução quando a0 >= 5
0x00159593	slli x11 x11 1	slli a1, a1, 1 # Realiza o shift para a esquerda de 1 bit no valor de a1
0x00150513	addi x10 x10 1	addi a0, a0, 1 # Incrementa o valor de a0 em 1 unidade
0xff5ff06f	jal x0 -12	j loop # Retorna para o loop quando a condição de parada não é atingida
0x00000013	addi x0 x0 0	fim: nop # Para a execução

console output

RegistersMemory

zero	0
ra (x1)	0
sp (x2)	2147483632
gp (x3)	268435456
tp (x4)	0
t0 (x5)	0
t1 (x6)	0
t2 (x7)	0
s0 (x8)	0
s1 (x9)	0
a0 (x10)	1
a1 (x11)	2
a2 (x12)	5
a3 (x13)	0

Ao final da execução do loop:

Run
Step
Prev
Reset
Dump

Machine Code	Basic Code	Original Code
0x00100513	addi x10 x0 1	li a0, 1 # a0 = 1
0x00a505b3	add x11 x10 x10	add a1, a0, a0 # a1 = 2 * a0
0x00500613	addi x12 x0 5	li a2, 5 # a2 = 5
0x00c50063	beq x10 x12 16	loop: beq a0, a2, fim # Para a execução quando a0 >= 5
0x00159593	slli x11 x11 1	slli a1, a1, 1 # Realiza o shift para a esquerda de 1 bit no valor de a1
0x00150513	addi x10 x10 1	addi a0, a0, 1 # Incrementa o valor de a0 em 1 unidade
0xff5ff06f	jal x0 -12	j loop # Retorna para o loop quando a condição de parada não é atingida
0x00000013	addi x0 x0 0	fim: nop # Para a execução

console output

Registers
Memory

zero	0
ra (x1)	0
sp (x2)	2147483632
gp (x3)	268435456
tp (x4)	0
t0 (x5)	0
t1 (x6)	0
t2 (x7)	0
s0 (x8)	0
s1 (x9)	0
a0 (x10)	5
a1 (x11)	32
a2 (x12)	5
a3 (x13)	0

PROBLEMA 2

a) O código em Assembly correspondente é:

```
addi x10 x0 2
addi x11 x0 4
beq x10 x11 12
add x10 x10 x10
jal x0 -8
add x12 x11 x11
addi x0 x0 0
```

Contudo, para que ele seja executado pelo Venus foi preciso alterá-lo para:

```
addi x10 x0 2
addi x11 x0 4

loop: beq x10 x11 fim
add x10 x10 x10
jal x0 loop

fim: add x12, x11, x11
addi x0, x0, 0
```

b) A princípio, o código faz a atribuição de 2 e 4 nos registradores x10 e x11, respectivamente. Em seguida, checa a condição de x10 ter o mesmo valor que x11. Se essa condição é verdadeira, o código armazena em x12 o valor de x11 multiplicado por 2 e encerra a execução. Se não for verdadeira, ele armazena em x10 o dobro do seu valor e repete a verificação da condição.

c) Os registradores utilizados no código foram o x0, x10, x11 e x12.

d) O conteúdos dos registradores podem ser verificados abaixo:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00200513	addi x10 x0 2	addi x10 x0 2
0x00400593	addi x11 x0 4	addi x11 x0 4
0x00b50663	beq x10 x11 12	loop: beq x10 x11 fim
0x00a50533	add x10 x10 x10	add x10 x10 x10
0xff9ff06f	jal x0 -8	jal x0 loop
0x00b50633	add x12 x11 x11	fim: add x12, x11, x11
0x00000013	addi x0 x0 0	addi x0, x0, 0

console output

RegistersMemory

zero

0x00000000

ra (x1)

0x00000000

sp (x2)

0x7fffffff0

gp (x3)

0x10000000

tp (x4)

0x00000000

t0 (x5)

0x00000000

t1 (x6)

0x00000000

t2 (x7)

0x00000000

s0 (x8)

0x00000000

s1 (x9)

0x00000000

a0 (x10)

0x00000004

a1 (x11)

0x00000004

a2 (x12)

0x00000000

a3 (x13)

0x00000000

e) Os prints estão dispostos abaixo:

Após a execução da primeira instrução:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00200513	addi x10 x0 2	addi x10 x0 2
0x00400593	addi x11 x0 4	addi x11 x0 4
0x00b50663	beq x10 x11 12	loop: beq x10 x11 fim
0x00a50533	add x10 x10 x10	add x10 x10 x10
0xff9ff06f	jal x0 -8	jal x0 loop
0x00b50633	add x12 x11 x11	fim: add x12, x11, x11
0x00000013	addi x0 x0 0	addi x0, x0, 0

console output

RegistersMemory

zero

0x00000000

ra (x1)

0x00000000

sp (x2)

0x7fffffff0

gp (x3)

0x10000000

tp (x4)

0x00000000

t0 (x5)

0x00000000

t1 (x6)

0x00000000

t2 (x7)

0x00000000

s0 (x8)

0x00000000

s1 (x9)

0x00000000

a0 (x10)

0x00000002

a1 (x11)

0x00000000

a2 (x12)

0x00000000

a3 (x13)

0x00000000

Após a execução da segunda instrução:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00200513	addi x10 x0 2	addi x10 x0 2
0x00400593	addi x11 x0 4	addi x11 x0 4
0x00b50663	beq x10 x11 12	loop: beq x10 x11 fim
0x00a50533	add x10 x10 x10	add x10 x10 x10
0xff9ff06f	jal x0 -8	jal x0 loop
0x00b50633	add x12 x11 x11	fim: add x12, x11, x11
0x00000013	addi x0 x0 0	addi x0, x0, 0

console output

RegistersMemory

zero

0x00000000

ra (x1)

0x00000000

sp (x2)

0x7fffffff0

gp (x3)

0x10000000

tp (x4)

0x00000000

t0 (x5)

0x00000000

t1 (x6)

0x00000000

t2 (x7)

0x00000000

s0 (x8)

0x00000000

s1 (x9)

0x00000000

a0 (x10)

0x00000002

a1 (x11)

0x00000004

a2 (x12)

0x00000000

a3 (x13)

0x00000000

Após a execução da terceira instrução:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00200513	addi x10 x0 2	addi x10 x0 2
0x00400593	addi x11 x0 4	addi x11 x0 4
0x00b50663	beq x10 x11 12	loop: beq x10 x11 fim
0x00a50533	add x10 x10 x10	add x10 x10 x10
0xff9ff06f	jal x0 -8	jal x0 loop
0x00b58633	add x12 x11 x11	fim: add x12, x11, x11
0x00000013	addi x0 x0 0	addi x0, x0, 0

console output

RegistersMemory

zero	0x00000000
ra (x1)	0x00000000
sp (x2)	0x7ffffff0
gp (x3)	0x10000000
tp (x4)	0x00000000
t0 (x5)	0x00000000
t1 (x6)	0x00000000
t2 (x7)	0x00000000
s0 (x8)	0x00000000
s1 (x9)	0x00000000
a0 (x10)	0x00000002
a1 (x11)	0x00000004
a2 (x12)	0x00000000
a3 (x13)	0x00000000

Após a execução da quarta instrução:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00200513	addi x10 x0 2	addi x10 x0 2
0x00400593	addi x11 x0 4	addi x11 x0 4
0x00b50663	beq x10 x11 12	loop: beq x10 x11 fim
0x00a50533	add x10 x10 x10	add x10 x10 x10
0xff9ff06f	jal x0 -8	jal x0 loop
0x00b58633	add x12 x11 x11	fim: add x12, x11, x11
0x00000013	addi x0 x0 0	addi x0, x0, 0

console output

RegistersMemory

zero	0x00000000
ra (x1)	0x00000000
sp (x2)	0x7ffffff0
gp (x3)	0x10000000
tp (x4)	0x00000000
t0 (x5)	0x00000000
t1 (x6)	0x00000000
t2 (x7)	0x00000000
s0 (x8)	0x00000000
s1 (x9)	0x00000000
a0 (x10)	0x00000004
a1 (x11)	0x00000004
a2 (x12)	0x00000000
a3 (x13)	0x00000000

Após a execução da quinta instrução:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00200513	addi x10 x0 2	addi x10 x0 2
0x00400593	addi x11 x0 4	addi x11 x0 4
0x00b50663	beq x10 x11 12	loop: beq x10 x11 fim
0x00a50533	add x10 x10 x10	add x10 x10 x10
0xff9ff06f	jal x0 -8	jal x0 loop
0x00b58633	add x12 x11 x11	fim: add x12, x11, x11
0x00000013	addi x0 x0 0	addi x0, x0, 0

console output

RegistersMemory

zero	0x00000000
ra (x1)	0x00000000
sp (x2)	0x7ffffff0
gp (x3)	0x10000000
tp (x4)	0x00000000
t0 (x5)	0x00000000
t1 (x6)	0x00000000
t2 (x7)	0x00000000
s0 (x8)	0x00000000
s1 (x9)	0x00000000
a0 (x10)	0x00000004
a1 (x11)	0x00000004
a2 (x12)	0x00000000
a3 (x13)	0x00000000

Após a execução da terceira instrução pela segunda vez:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00200513	addi x10 x0 2	addi x10 x0 2
0x00400593	addi x11 x0 4	addi x11 x0 4
0x00b50663	beq x10 x11 12	loop: beq x10 x11 fim
0x00a50533	add x10 x10 x10	add x10 x10 x10
0xff9ff06f	jal x0 -8	jal x0 loop
0x00b58633	add x12 x11 x11	fim: add x12, x11, x11
0x00000013	addi x0 x0 0	addi x0, x0, 0

console output

RegistersMemory

zero	0x00000000
ra (x1)	0x00000000
sp (x2)	0x7ffffff0
gp (x3)	0x10000000
tp (x4)	0x00000000
t0 (x5)	0x00000000
t1 (x6)	0x00000000
t2 (x7)	0x00000000
s0 (x8)	0x00000000
s1 (x9)	0x00000000
a0 (x10)	0x00000004
a1 (x11)	0x00000004
a2 (x12)	0x00000000
a3 (x13)	0x00000000

Após a execução da sexta instrução:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00200513	addi x10 x0 2	addi x10 x0 2
0x00400593	addi x11 x0 4	addi x11 x0 4
0x00b50663	beq x10 x11 12	loop: beq x10 x11 fim
0x00a50533	add x10 x10 x10	add x10 x10 x10
0xff9ff06f	jal x0 -8	jal x0 loop
0x00b58633	add x12 x11 x11	fim: add x12, x11, x11
0x00000013	addi x0 x0 0	addi x0, x0, 0

console output

RegistersMemory

zero	0x00000000
ra (x1)	0x00000000
sp (x2)	0x7ffffff0
gp (x3)	0x10000000
tp (x4)	0x00000000
t0 (x5)	0x00000000
t1 (x6)	0x00000000
t2 (x7)	0x00000000
s0 (x8)	0x00000000
s1 (x9)	0x00000000
a0 (x10)	0x00000004
a1 (x11)	0x00000004
a2 (x12)	0x00000008
a3 (x13)	0x00000000

Após a execução da sétima instrução:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x00200513	addi x10 x0 2	addi x10 x0 2
0x00400593	addi x11 x0 4	addi x11 x0 4
0x00b50663	beq x10 x11 12	loop: beq x10 x11 fim
0x00a50533	add x10 x10 x10	add x10 x10 x10
0xff9ff06f	jal x0 -8	jal x0 loop
0x00b58633	add x12 x11 x11	fim: add x12, x11, x11
0x00000013	addi x0 x0 0	addi x0, x0, 0

console output

RegistersMemory

zero	0x00000000
ra (x1)	0x00000000
sp (x2)	0x7ffffff0
gp (x3)	0x10000000
tp (x4)	0x00000000
t0 (x5)	0x00000000
t1 (x6)	0x00000000
t2 (x7)	0x00000000
s0 (x8)	0x00000000
s1 (x9)	0x00000000
a0 (x10)	0x00000004
a1 (x11)	0x00000004
a2 (x12)	0x00000008
a3 (x13)	0x00000000