Problema 1 - Código em Assembly

```
.text
main:
        addi a0, zero, 1
        add a1, a0, a0
        addi a2, zero, 5
loop:
        beq a0, a2, fim
        slli a1, a1, 1
        addi a0, a0, 1
        j loop
fim:
        nop
```

a) Qual é a operação realizada pelo código acima?

Ele executa um loop que dobra um valor (inicialmente 2) até um contador alcançar um valor determinado (neste caso, 5).

Inicializa a0 com 1, a1 com 2 e a2 com 5. Em seguida, entra em um loop onde dobra o valor de a1 e incrementa o valor de a0 até que a0 seja igual a a2. Então, o loop termina e o programa alcança a instrução nop. Portanto, no final da execução do programa, a0 será 5, a1 será 32 (2 dobrado quatro vezes) e a2 permanecerá como 5.

b) Qual é o conteúdo (em decimal) dos registradores a0, a1 e a2 ao final da execução do programa?

Na quinta iteração, a0 será igual a a2, o que causará a saída do loop. Portanto, no final da execução, os valores serão a0=5, a1=32 e a2=5.

Acompanhando as operações passo a passo, sabendo que o programa começa com a0=1, a1=2 e a2=5:

Iteração 1: a0=2, a1=4, a2=5 (valor em a1 é duplicado, a0 incrementado)

Iteração 2: a0=3, a1=8, a2=5 (valor em a1 é duplicado, a0 incrementado)

Iteração 3: a0=4, a1=16, a2=5 (valor em a1 é duplicado, a0 incrementado)

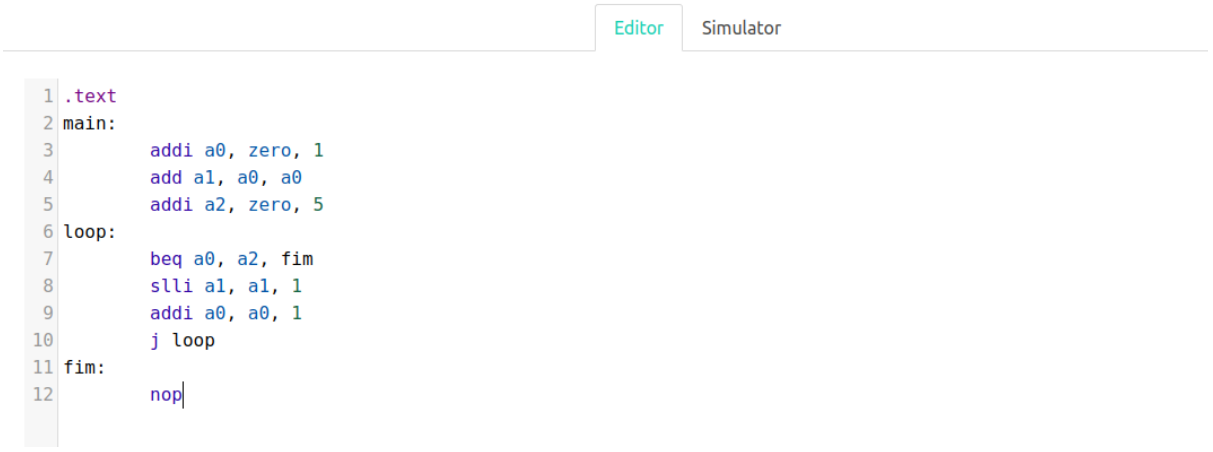
Iteração 4: a0=5, a1=32, a2=5 (valor em a1 é duplicado, a0 incrementado)

c) Quais são os valores de loop (na instrução j loop) e fim (na instrução beq a0, a2, fim)

fim = 16

loop = -12

d) Adicione "prints" da tela do simulador ao arquivo PDF. Esses "prints" deverão mostrar todo o ambiente, incluindo o código em Assembly que foi executado e os valores exibidos na interface após a execução de cada instrução (código de máquina armazenado na memória, valores do pc e dos demais registradores envolvidos).



Run | Step | Prev | Reset | Dump

Machine Code	Basic Code	Original Code
0x00100513	addi x10 x0 1	addi a0, zero, 1
0x00a505b3	add x11 x10 x10	add a1, a0, a0
0x00500613	addi x12 x0 5	addi a2, zero, 5
0x00c50863	beq x10 x12 16	beq a0, a2, fim
0x00159593	slli x11 x11 1	slli a1, a1, 1
0x00150513	addi x10 x10 1	addi a0, a0, 1
0xff5ff06f	jal x0 -12	j loop
0x00000013	addi x0 x0 0	nop

Registers | Memory

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)	0x00000005
a1 (x11)	0x00000020
a2 (x12)	0x00000005
a3 (x13)	0x00000000
a4 (x14)	0x00000000
a5 (x15)	0x00000000
a6 (x16)	0x00000000

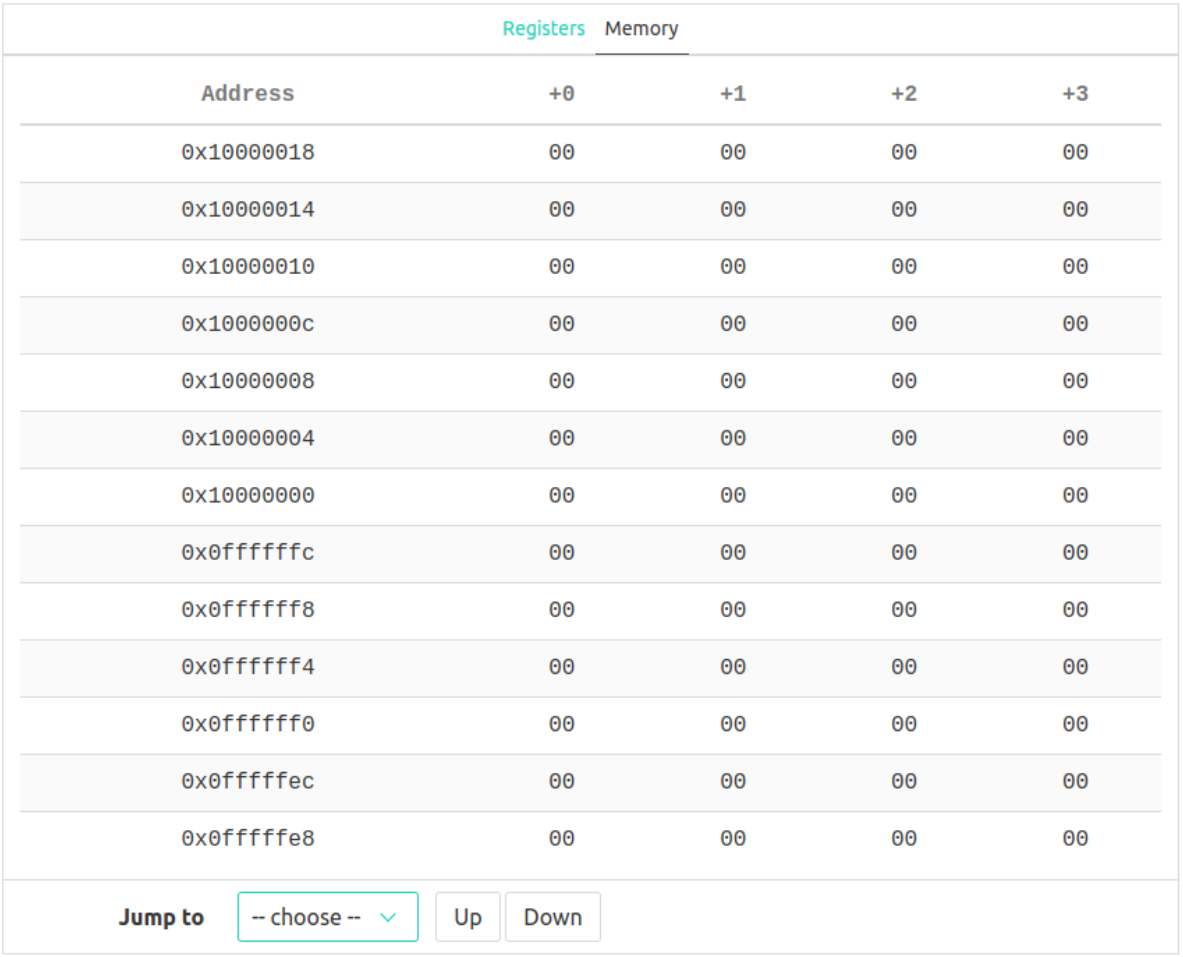
a6 (x16)	0x00000000
a7 (x17)	0x00000000
s2 (x18)	0x00000000
s3 (x19)	0x00000000
s4 (x20)	0x00000000
s5 (x21)	0x00000000
s6 (x22)	0x00000000
s7 (x23)	0x00000000
s8 (x24)	0x00000000
s9 (x25)	0x00000000
s10 (x26)	0x00000000
s11 (x27)	0x00000000
t3 (x28)	0x00000000
t4 (x29)	0x00000000
t5 (x30)	0x00000000
t6 (x31)	0x00000000

Display Settings Hex

Registers Memory

Address	+0	+1	+2	+3
0x00000018	6f	f0	5f	ff
0x00000014	13	05	15	00
0x00000010	93	95	15	00
0x0000000c	63	08	c5	00
0x00000008	13	06	50	00
0x00000004	b3	05	a5	00
0x00000000	13	05	10	00
----------	--	--	--	--
----------	--	--	--	--
----------	--	--	--	--
----------	--	--	--	--
----------	--	--	--	--
----------	--	--	--	--

Jump to -- choose -- Up Down



Registers Memory

Address	+0	+1	+2	+3
0x10000018	00	00	00	00
0x10000014	00	00	00	00
0x10000010	00	00	00	00
0x1000000c	00	00	00	00
0x10000008	00	00	00	00
0x10000004	00	00	00	00
0x10000000	00	00	00	00
0x0ffffffc	00	00	00	00
0x0ffffff8	00	00	00	00
0x0ffffff4	00	00	00	00
0x0ffffff0	00	00	00	00
0x0fffffec	00	00	00	00
0x0fffffe8	00	00	00	00

Jump to -- choose -- Up Down

Problema 2 - Código de Máquina

a) Qual é o código em Assembly correspondente?

```
 main:
    addi x10, x0, 2
    addi x11, x0, 4
loop:
    beq x10, x11, end
    add x10, x10, x10
    j loop
end:
    add x12, x11, x11
    addi x0, x0, 0
```

b) Qual é a operação realizada pelo código acima?

Inicialmente, o código define dois registradores (x10 e x11) para os valores 2 e 4, respectivamente. Ele então entra em um loop onde verifica se x10 é igual a x11. Se for, o programa vai para o fim do código. Se não for, ele dobra o valor de x10 (adicionando x10 a si mesmo) e repete o loop. Quando x10 é igual a x11, o programa sai do loop e calcula o dobro do valor em x11, armazenando-o em x12. Finalmente, ele define o registrador x0 (ou zero) para zero, que já é o valor padrão para x0 no RISC-V.

c) Quais são os registradores utilizados no código?

Os registradores utilizados no código são x10, x11, x12 e x0.

d) Quais são os conteúdos desses registradores ao final da execução do programa

x10: O valor de x10 será 4, o mesmo que x11, porque esse é o ponto em que o loop termina.

x11: O valor de x11 permanecerá 4, porque é apenas lido e nunca alterado no programa.

x12: O valor de x12 será 8, que é o dobro do valor final de x11.

x0: O valor de x0 será 0, porque x0 é o registrador zero em RISC-V, que é definido para ser sempre zero e qualquer tentativa de escrever nele será ignorada.

e) Adicione "prints" da tela do simulador ao arquivo PDF. Esses "prints" deverão mostrar todo o ambiente, incluindo o código em Assembly que foi executado e os valores exibidos na interface após a execução de cada instrução (código de máquina armazenado na memória, valores do pc e dos demais registradores envolvidos).

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	beq x10, x11, end
0x00a50533	add x10 x10 x10	add x10, x10, x10
0xff9ff06f	jal x0 -8	j loop
0x00b58633	add x12 x11 x11	add x12, x11, x11
0x00000013	addi x0 x0 0	addi x0, x0, 0

Registers | Memory

Address	+0	+1	+2	+3
0x00000030	00	00	00	00
0x0000002c	00	00	00	00
0x00000028	00	00	00	00
0x00000024	00	00	00	00
0x00000020	00	00	00	00
0x0000001c	00	00	00	00
0x00000018	13	00	00	00
0x00000014	33	86	b5	00
0x00000010	6f	f0	9f	ff
0x0000000c	33	05	a5	00
0x00000008	63	06	b5	00
0x00000004	93	05	40	00
0x00000000	13	05	20	00

Jump to -- choose -- Up Down

Registers Memory

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
a4 (x14)	0x00000000
a5 (x15)	0x00000000
a6 (x16)	0x00000000

Display Settings Hex

a7 (x17)	0x00000000
s2 (x18)	0x00000000
s3 (x19)	0x00000000
s4 (x20)	0x00000000
s5 (x21)	0x00000000
s6 (x22)	0x00000000
s7 (x23)	0x00000000
s8 (x24)	0x00000000
s9 (x25)	0x00000000
s10 (x26)	0x00000000
s11 (x27)	0x00000000
t3 (x28)	0x00000000
t4 (x29)	0x00000000
t5 (x30)	0x00000000
t6 (x31)	0x00000000

Display Settings Hex