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.

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:

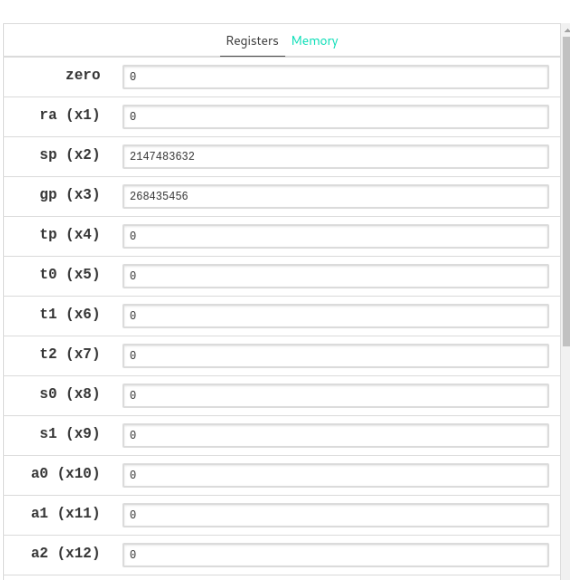
Print do código no editor:

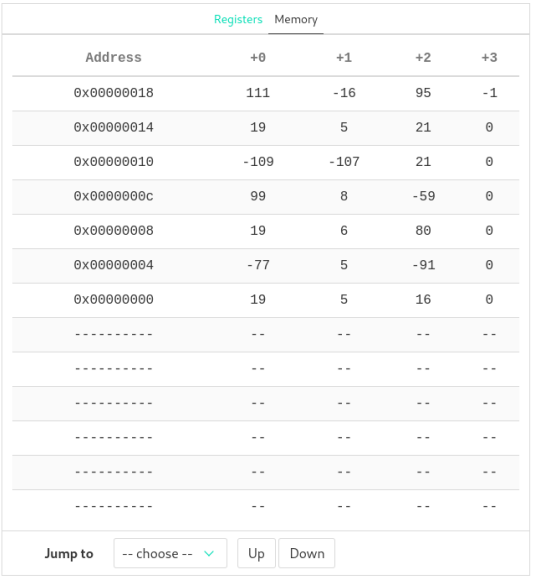
```
1         li a0, 1            # a0 = 1
2         add a1, a0, a0      # a1 = 2 * a0
3         li a2, 5            # a2 = 5
4
5  loop:  beq a0, a2, fim     # Para a execução quando a0 >= 5
6         slli a1, a1, 1      # Realiza o shift para a esquerda de 1 bit no valor de a1
7         addi a0, a0, 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:





Na execução da primeira linha de código:

Na execução da segunda linha:

Na execução da terceira linha de código:

Ao final da execução do loop:

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
```

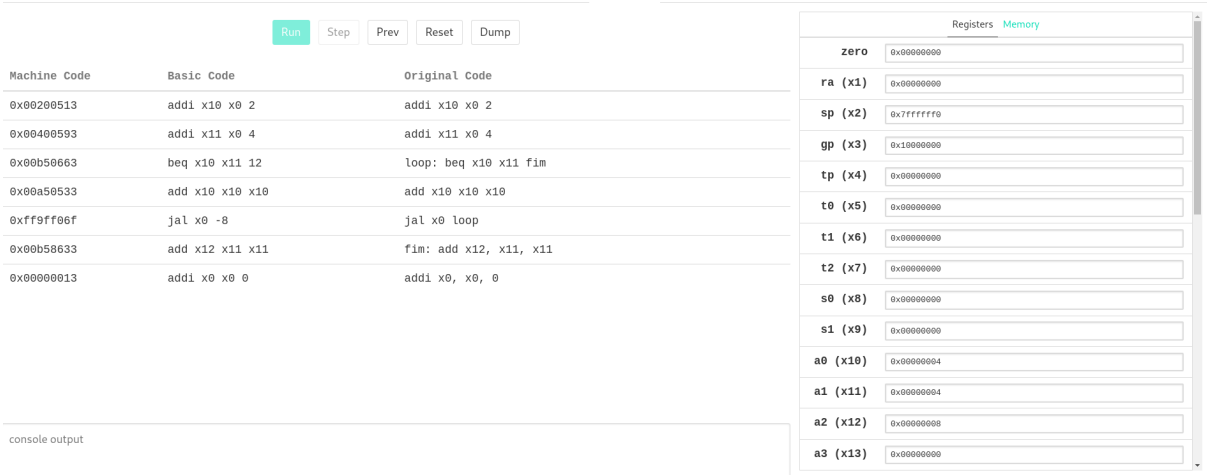
Contudo, para que ele seja executado pelo Venus foi preciso alterá-lo para:

```
        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.

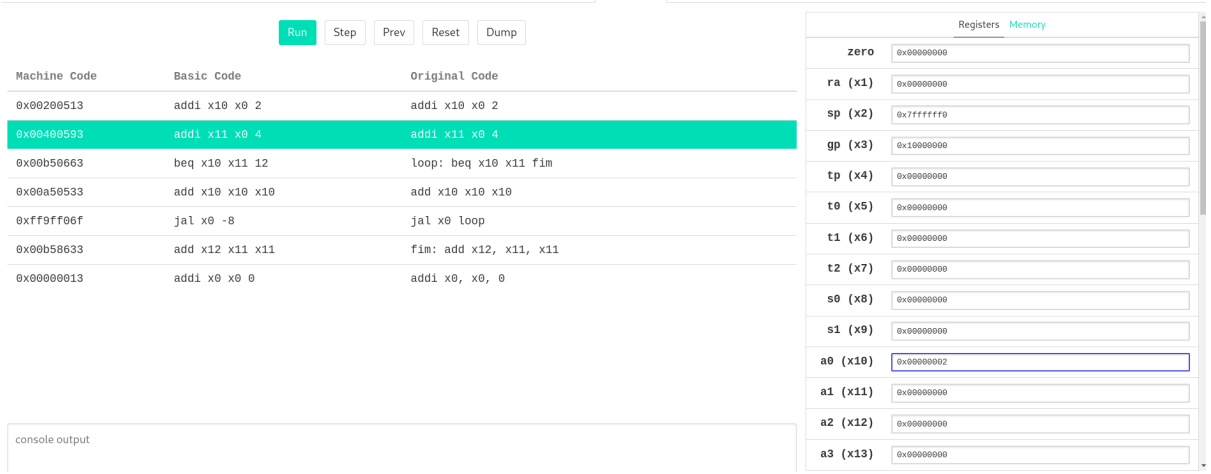
c) Os registradores utilizados no código foram o x0, x10, x11 e x12.

d) O conteúdos dos registradores podem ser verificados abaixo:

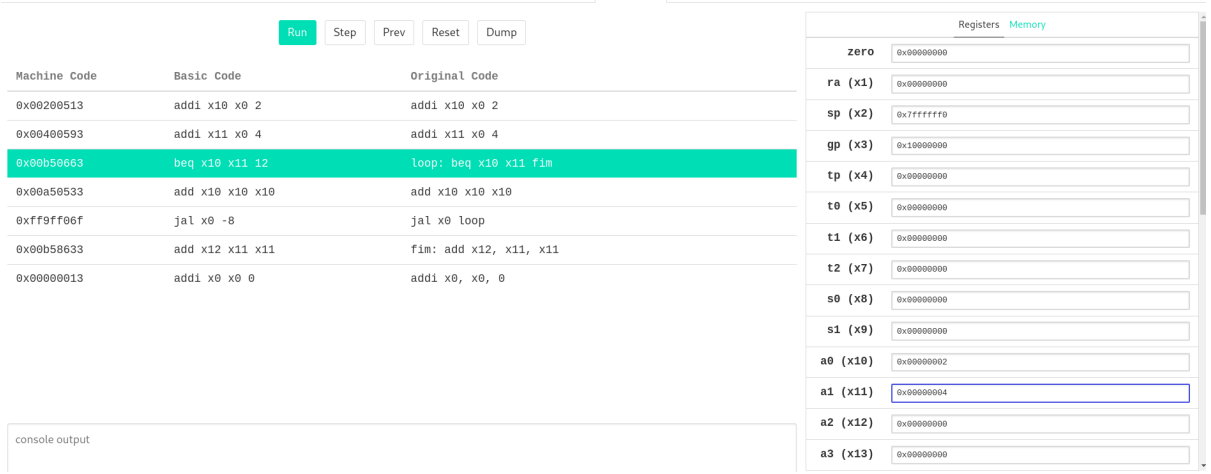


e) Os prints estão dispostos abaixo:

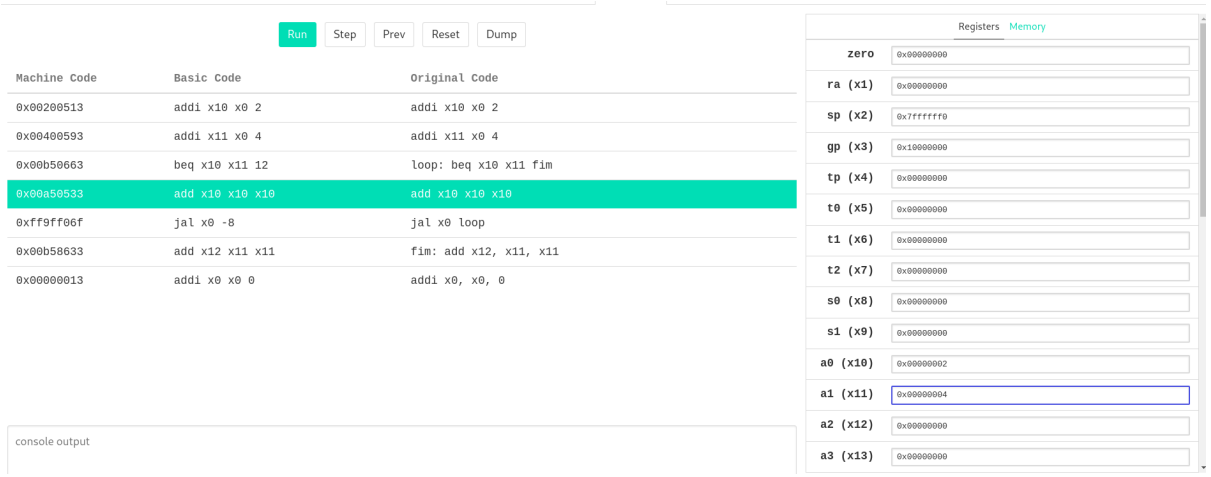
Após a execução da primeira instrução:



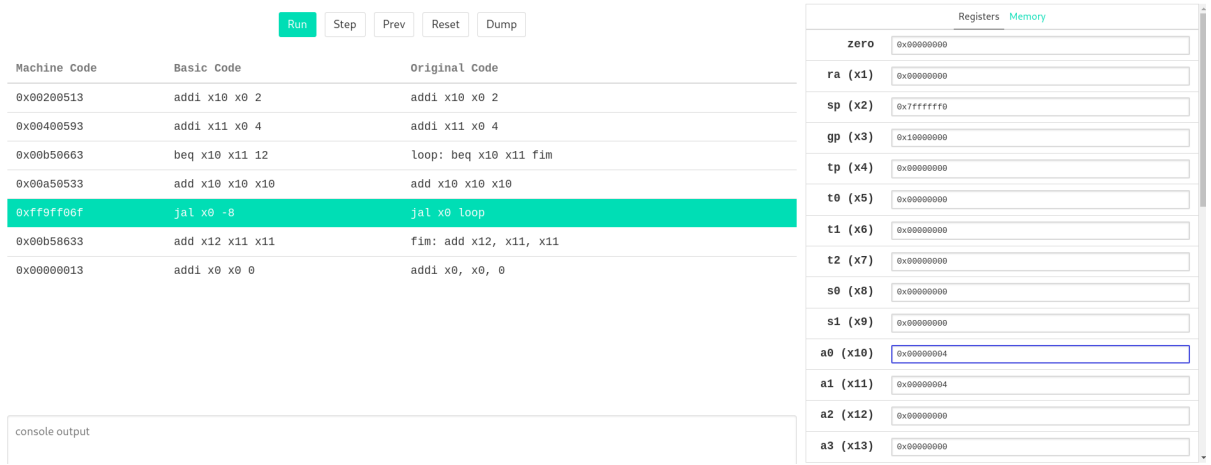
Após a execução da segunda instrução:



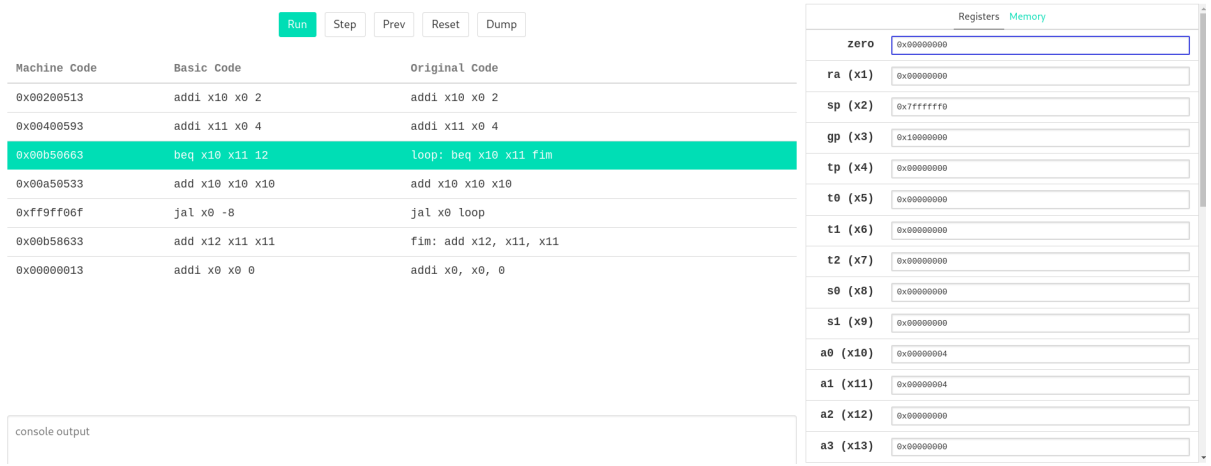
Após a execução da terceira instrução:



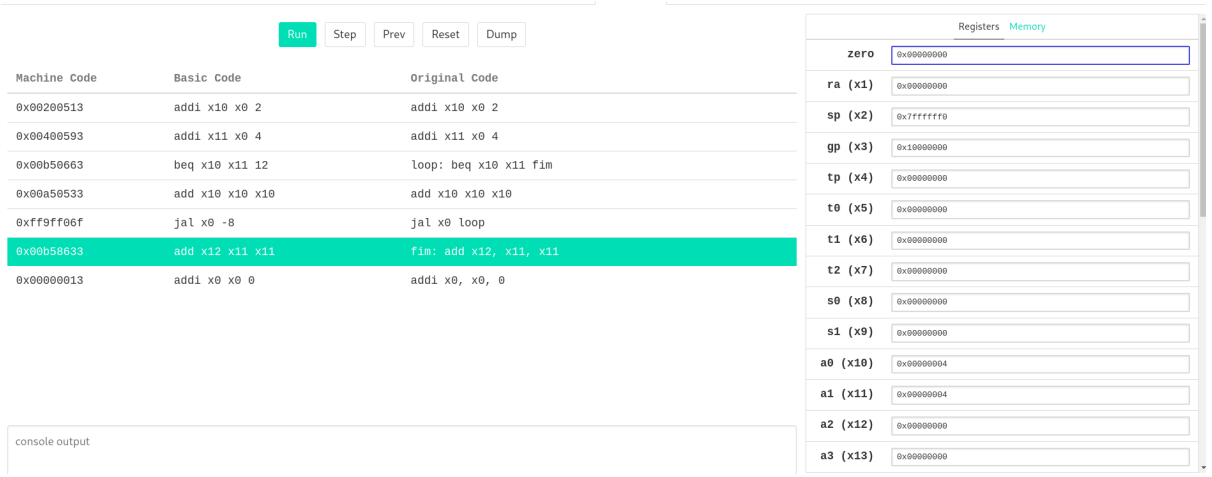
Após a execução da quarta instrução:



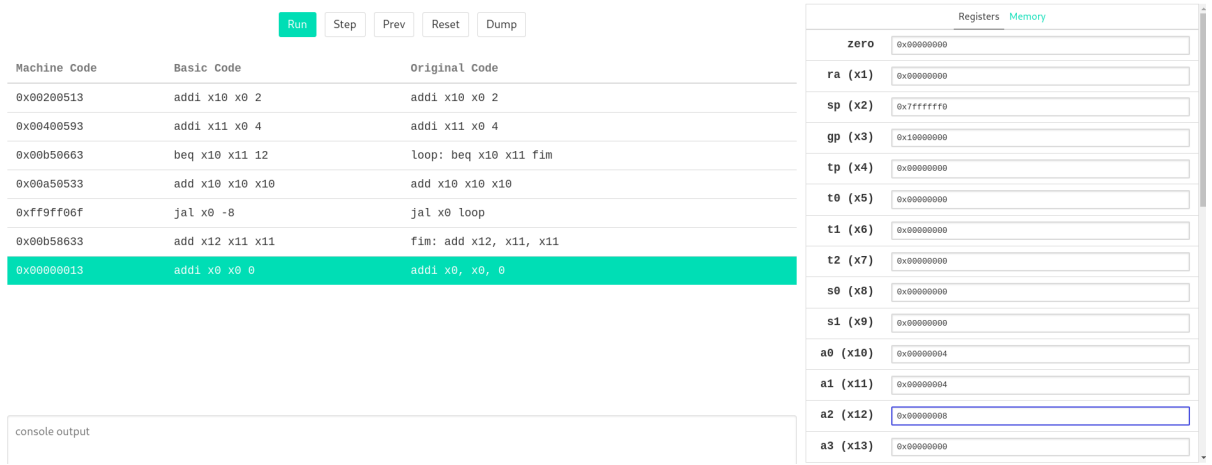
Após a execução da quinta instrução:



Após a execução da terceira instrução pela segunda vez:



Após a execução da sexta instrução:



Após a execução da sétima instrução: