

### UNIVERSIDADE FEDERAL DE CAMPINA GRANDE CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

### THAYANE STHEFFANY SILVA BARROS

**ROTEIRO 6** 

**OBS:** As imagens também estão disponíveis no outro arquivo enviado no arquivo compactado

## Problema 1 - Código em Assembly

```
Editor Simulator
1 .text
2 main:
3
           addi a0, zero, 1
4
           add al, a0, a0
           addi a2, zero, 5
6 loop;
           beq a0, a2, fim
8
           slli al, al, 1
9
           addi a0, a0, 1
10
           j loop
11 fim:
12
           nop
```

### a) Qual é a operação realizada pelo código acima?

A operação realizada pelo código é um loop que multiplica o valor em a1 por 2 (slli a1, a1, 1) enquanto o valor em a0 é incrementado (addi a0, a0, 1) até que a0 seja igual a 5 (beq a0, a2, fim).

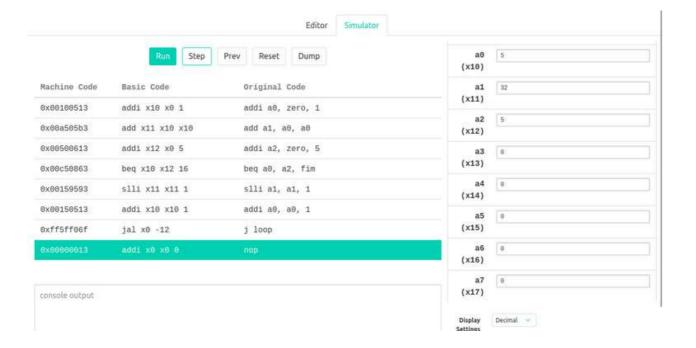
# b) Qual é o conteúdo (em decimal) dos registradores a0, a1 e a2 ao final da execução do programa?

Ao final da execução do programa, o conteúdo dos registradores é o seguinte:

a0: 5 (decimal)

• a1: 32 (decimal)

a2: 5 (decimal)



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

Os valores de "loop" e "fim" são rótulos (endereços de memória) para as instruções correspondentes. Portanto, eles não possuem valores numéricos específicos.

No entanto, em relação à posição de memória podemos determinar que o valor de "loop" é -12 (ou 0xff5ff06f em hexadecimal). Isso pode ser visto na instrução "jal x0 -12", onde o deslocamento é um valor negativo que representa a diferença entre o endereço atual da instrução e o endereço da instrução rotulada "loop". Nesse caso, o deslocamento de -12 indica que o endereço da instrução "loop" é 12 bytes (3 instruções) atrás do endereço atual.

O valor de "fim" é 16 (ou 0x00c50863 em hexadecimal). Isso pode ser visto na instrução "beq x10 x12 16", onde o deslocamento é um valor positivo que representa a diferença entre o endereço atual da instrução e o endereço da instrução rotulada "fim". Nesse caso, o deslocamento de 16 indica que o endereço da instrução "fim" é 16 bytes (4 instruções) à frente do endereço atual. Portanto:

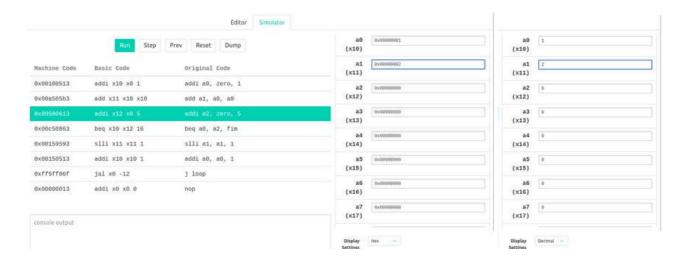
- O valor de "loop" é -12.
- O valor de "fim" é 16.
- d) Adicione "prints" da tela do simulador ao arquivo PDF. Esses "prints" deverão mostrar todo o ambiente, incluindo o código em Assembly que foi executado e os valores exibidos na interface após a execução de cada instrução (código de máquina armazenado na memória, valores do pc e dos demais registradores envolvidos).



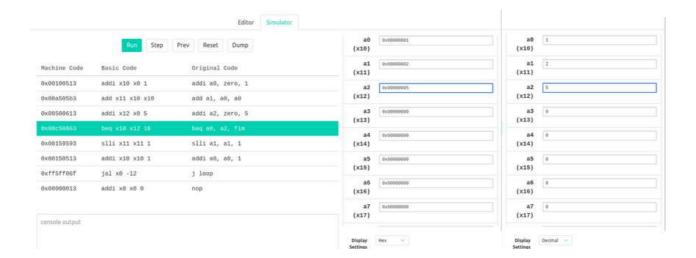
• Registrador a0 é carregado com o valor 1 (addi a0, zero, 1).



 Registrador a1 é carregado com o valor em a0 adicionado a a0, resultando em 2 (add a1, a0, a0).



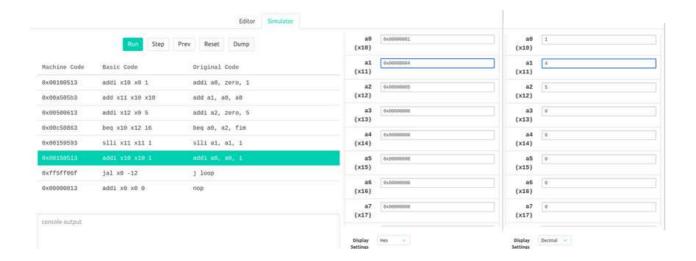
• Registrador a2 é carregado com o valor 5 (addi a2, zero, 5).



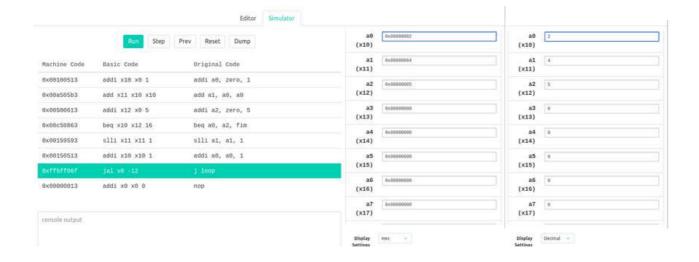
 Verifica se o valor em a0 é igual ao valor em a2. Como não são (a0 = 1 e a2 = 5) o programa continua a execução (beq a0, a2, fim).



• O valor em a1 é deslocado uma posição à esquerda, multiplicando-o por 2 (slli a1, a1, 1).



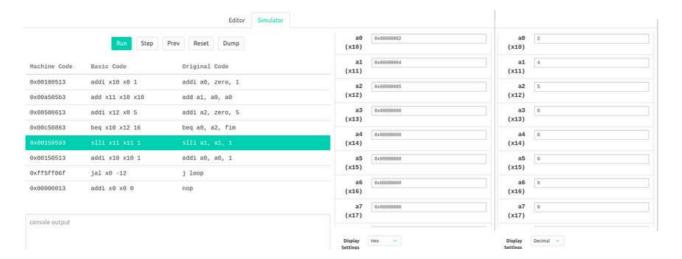
• O valor em a0 é incrementado em 1 (addi a0, a0, 1).



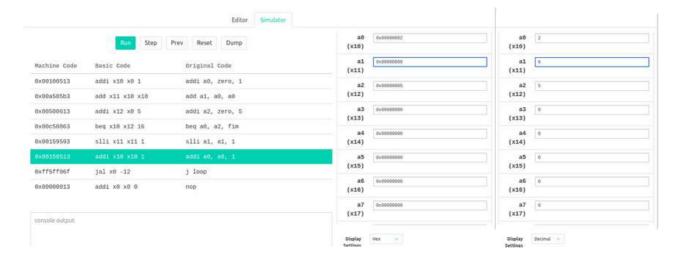
• O programa salta de volta para a instrução loop para repetir o processo.



Verifica-se novamente se o valor em a0 é igual ao valor em a2. Como não são (a0 = 2 e a2 = 5) o programa continua a execução (beq a0, a2, fim).



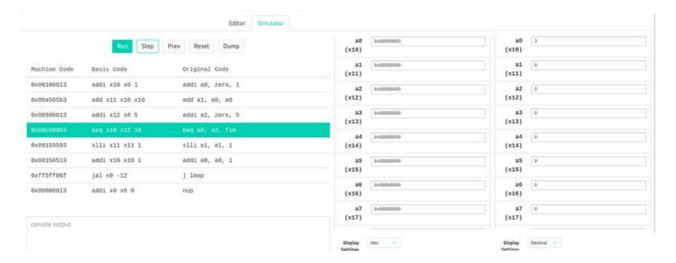
• O valor em a1 é deslocado uma posição à esquerda, multiplicando-o por 2 (slli a1, a1, 1).



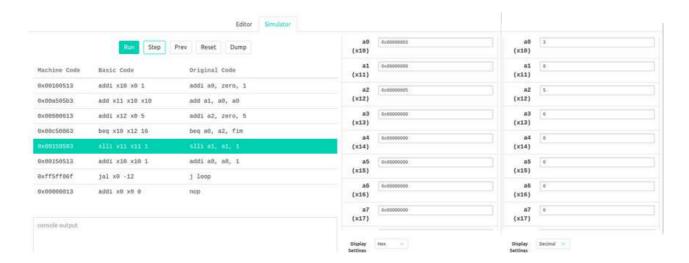
• O valor em a0 é incrementado em 1 (addi a0, a0, 1). a0 agora tem o valor 3.



• O programa salta de volta para a instrução loop para repetir o processo.



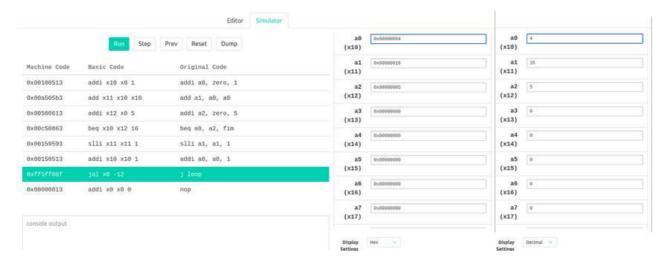
Verifica-se novamente se o valor em a0 é igual ao valor em a2. Como não são (a0 = 3 e a2
 = 5) o programa continua a execução (beq a0, a2, fim).



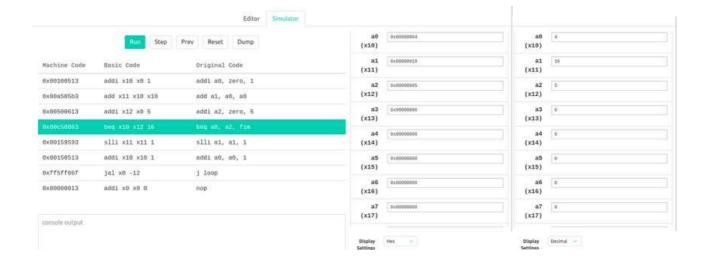
• O valor em a1 é deslocado uma posição à esquerda, multiplicando-o por 2 (slli a1, a1, 1).



• O valor em a0 é incrementado em 1 (addi a0, a0, 1). a0 agora tem o valor 4.



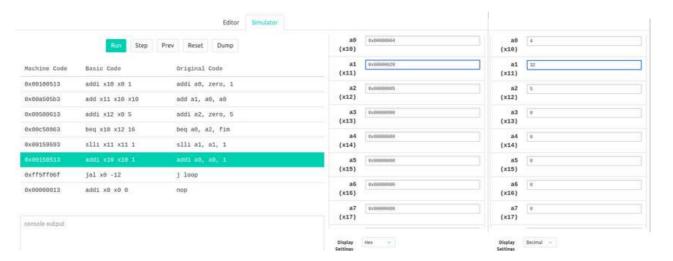
• O programa salta de volta para a instrução loop para repetir o processo.



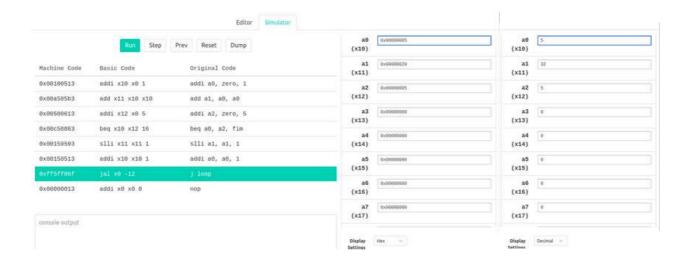
Verifica-se novamente se o valor em a0 é igual ao valor em a2. Como não são (a0 = 4 e a2
 = 5) o programa continua a execução (beq a0, a2, fim).



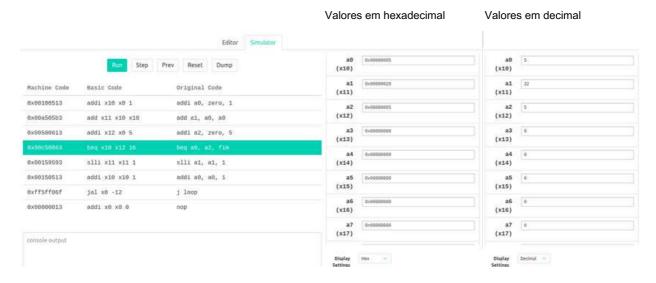
• O valor em a1 é deslocado uma posição à esquerda, multiplicando-o por 2 (slli a1, a1, 1).



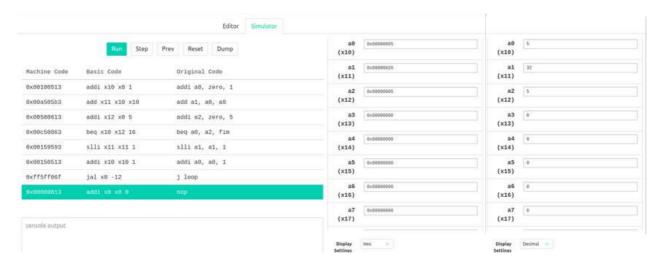
• O valor em a0 é incrementado em 1 (addi a0, a0, 1). a0 agora tem o valor 5.



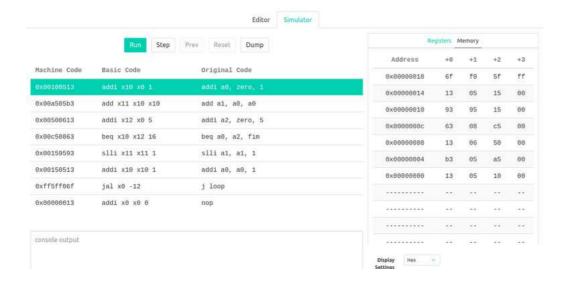
 O programa salta de volta para a instrução loop para repetir o processo. Verifica se o valor em a0 é igual ao valor em a2. Como são iguais (a0 = 5 e a2 = 5), o programa salta para a instrução fim.



 A instrução nop não realiza nenhuma operação. É usada como uma instrução vazia para fins de controle do fluxo de execução.



Print da memória



### Problema 2 - Código de Máquina

#### a) Qual é o código em Assembly correspondente?

Instruções em hexadecimal: Instruções em Binário Instrução em RV32I

0x00200513: 0000 0000 0010 0000 0000 0101 0**001 0011** : **ADDI** 0000 0000 0100 0000 0000 0101 1**001 0011** : **ADDI** 0x00400593: 0000 0000 1011 0101 0000 0110 0110 0011 : **BEQ** 0x00b50663: 0000 0000 1010 0101 0000 0101 0**011 0011** : **ADD** 0x00a50533: 1111 1111 1001 1111 1111 0000 0**110 1111** : 0xff9ff06f: JAL 0000 0000 1011 0101 1000 0110 0**011 0011** : 0x00b58633: **ADD** 0000 0000 0000 0000 0000 0000 0**001 0011** : 0x00000013: NOP

Montando o código através das instruções em binário

**0000 0000 0010** 0000 0000 0101 0001 0011

**2** 0 10 (a0) : ADDI X10, X0, 2

**0000 0000 0100** 0000 0000 0101 1001 0011

4 0 11 (a1) : ADDI X11, X0, 4

**0000 000**0 1011 0101 0000 **0110 0**110 0011

11 10 - 0000 0000 0110 0 = 12 : BEQ X10, X11, 12

0000 0000 1010 0101 0000 0101 0011 0011

10 10 : ADD X10, X10, X10

**1111 1111 1001 1111 1111** 0000 0110 1111

• 1111 1111 1111 11100 = - 4 : JAL X0, -4

0000 0000 1011 0101 1000 **0110 0**011 0011

11 11 12 : ADD X12, X11, X11

0000 0000 0000 0000 0000 0000 0001 0011 : NOP

### O código convertido para a linguagem assembly RISC-V:

```
.text
main:
    ADDI a0, zero, 2
    ADDI a1, zero, 4
loop:
    BEQ a0, a1, fim
    ADD a0, a0, a0
    JAL zero, loop
fim:
    ADD a2, a1, a1
    NOP
```

### b) Qual é a operação realizada pelo código acima?

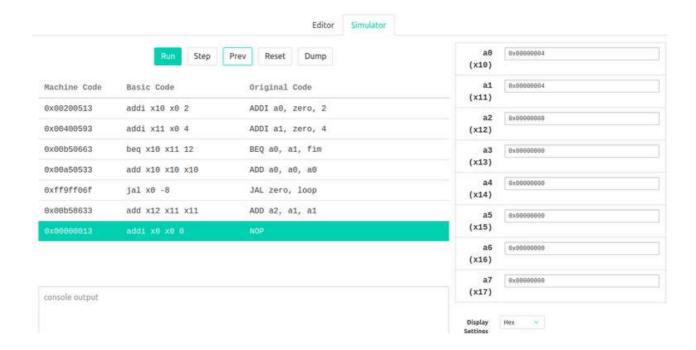
É um loop que multiplica o valor em a0 por 2 repetidamente até que a0 seja igual a a1. Em cada iteração, o valor em a0 é duplicado pela instrução ADD a0, a0, a0, e então o programa salta de volta para o rótulo "loop" usando a instrução JAL zero, loop.

### c) Quais são os registradores utilizados no código?

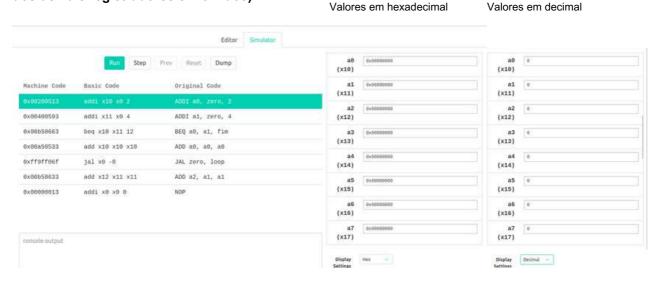
Os registradores utilizados no código são zero, a0, a1 e a2.

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

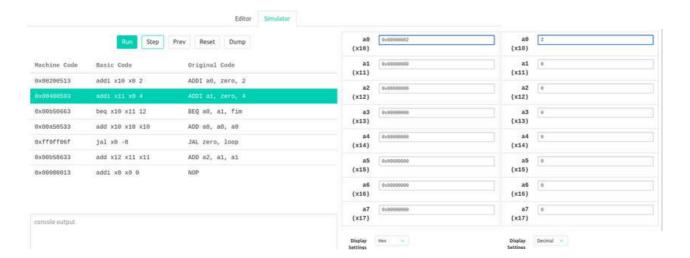
- O conteúdo do registrador a0 será igual ao conteúdo inicial do resgistrador a1 ou seja 4.
- O conteúdo do registrador a1 permanecerá o mesmo (4).
- O conteúdo do registrador a2 será igual a 2\*a1, ou seja, 8.



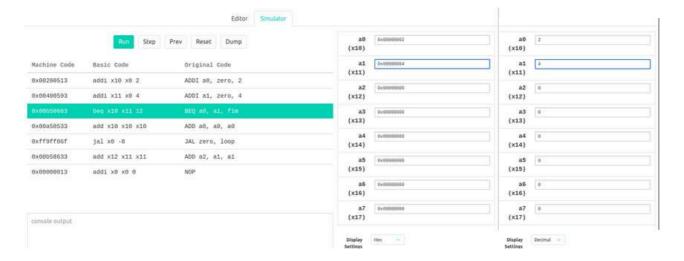
e) Adicione "prints" da tela do simulador ao arquivo PDF. Esses "prints" deverão mostrar todo o ambiente, incluindo o código em Assembly que foi executado e os valores exibidos na interface após a execução de cada instrução (código de máquina armazenado na memória, valores do pc e dos demais registradores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



• Registrador a0 é carregado com o valor 2 (ADDI a0, zero, 2).



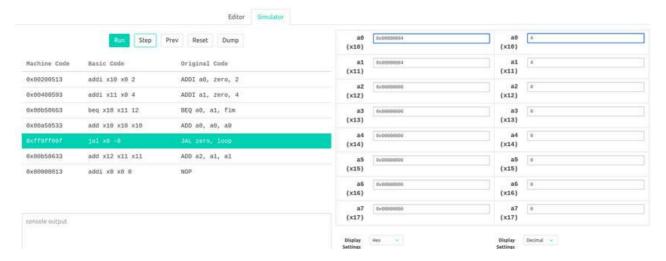
Registrador a1 é carregado com o valor 4 (ADDI a1, zero, 4).



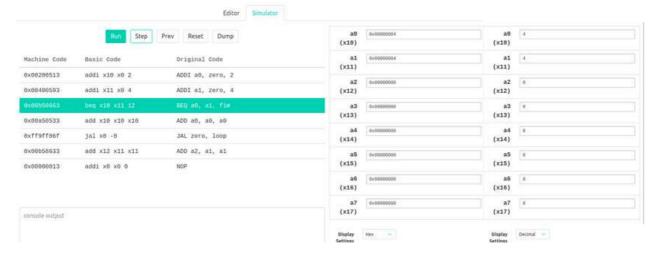
- O programa entra em um loop. Verifica se o valor em a0 é igual ao valor em a1 usando a instrução BEQ a0, a1, fim.
- Como são diferentes (a0 = 2 e a1 = 4), o programa executa a instrução ADD a0, a0, a0 para multiplicar o valor em a0 por 2.



Registrador a0 é carregado com o valor 4 (ADD a0, a0, a0).



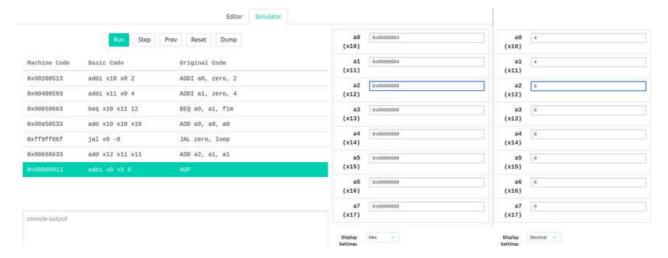
 O programa salta de volta para a instrução loop para repetir o processo usando a instrução JAL zero, loop



 O programa salta de volta para a instrução loop para repetir o processo. Verifica se o valor em a0 é igual ao valor em a1. Como são iguais (a0 = 4 e a1 = 4), o programa salta para a instrução fim.



- Quando o programa salta para o rótulo "fim", executa a instrução ADD a2, a1, a1 para duplicar o valor em a1 e armazena o resultado em a2.
- Registrador a2 é carregado com o valor 8.



- Finalmente, o programa executa a instrução NOP que não realiza nenhuma operação. É usada como uma instrução vazia para fins de controle do fluxo de execução.
- Print da memória:

