Exercício 02

Nome do aluno:

Fábio Volkmann Coelho

Objetivo

Aprender a manipular vetores e endereços de memória utilizando a linguagem de montagem RISC-V.

Instruções

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

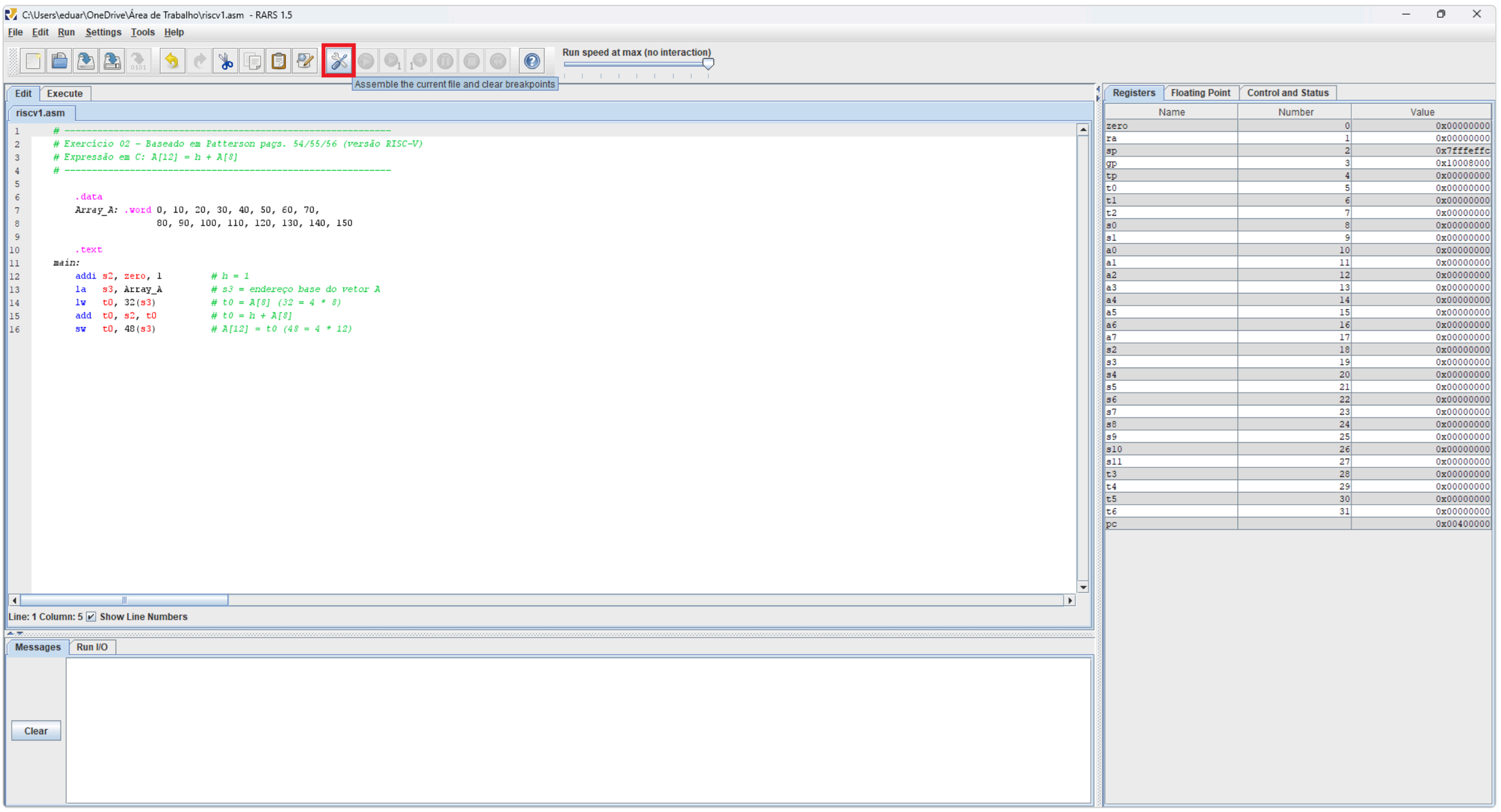
```
# ------------------------------------------------------------
# Exercício 02 - Baseado em Patterson pags. 54/55/56 (versão RISC-V)
# Expressão em C: A[12] = h + A[8]
# ------------------------------------------------------------

    .data
    Array_A: .word 0, 10, 20, 30, 40, 50, 60, 70,
                   80, 90, 100, 110, 120, 130, 140, 150

    .text
main:
    addi s2, zero, 1         # h = 1
    la   s3, Array_A         # s3 = endereço base do vetor A
    lw   t0, 32(s3)          # t0 = A[8] (32 = 4 * 8)
    add  t0, s2, t0          # t0 = h + A[8]
    sw   t0, 48(s3)          # A[12] = t0 (48 = 4 * 12)
```

O vetor **Array_A** possui 16 inteiros, cada um ocupando 4 bytes, armazenados sequencialmente na memória. O endereço inicial é atribuido pelo montador, em **0x10010000**.

Observe que a instrução **la** é uma pseudo-instrução traduzida pelo montador para duas instruções reais: **auipc** e **addi**, responsáveis por carregar o endereço base no registrador.

Montagem e Execução

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



Após a montagem, é possível visualizar o conteúdo das regiões de memória por meio da janela de segmentos. Nela, o usuário pode alternar entre:

- **Segmento de dados (.data):** região onde ficam armazenadas variáveis como vetores e constantes. Endereço inicial:

  `0x10010000 (.data)`

  .
- **Segmento da pilha (sp):** utilizado para chamadas de função e variáveis locais. Pode ser identificado como

  `0x7FFFEFE0 (sp)`

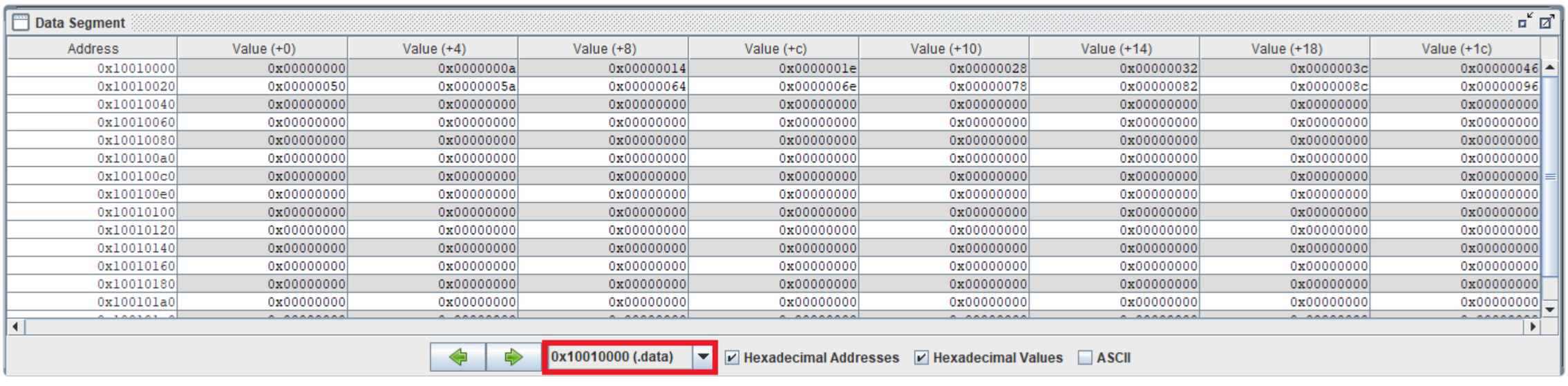
  .
- **Segmento de texto (.text):** contém as instruções do programa. Endereço inicial:

  `0x00400000 (.text)`

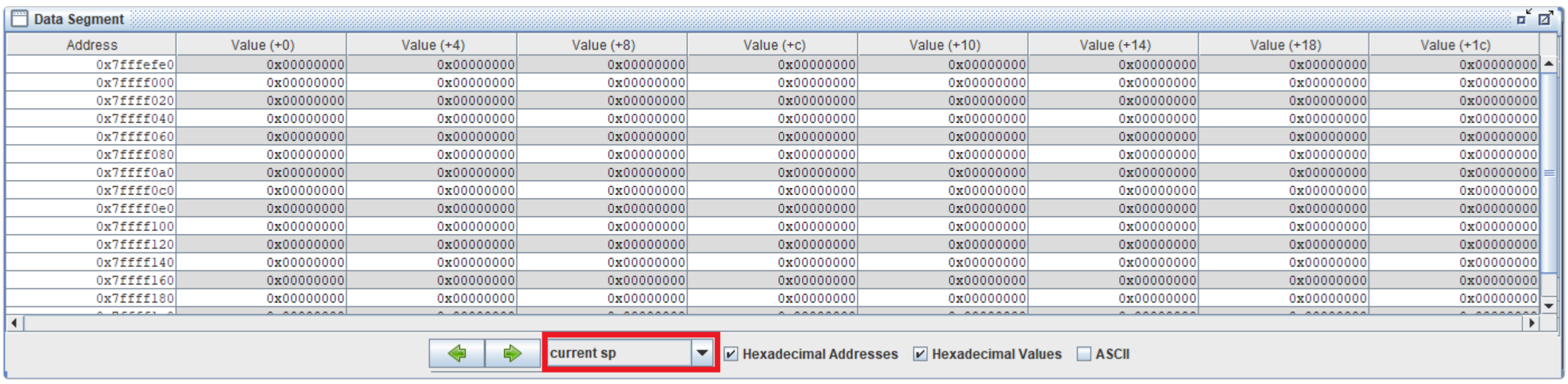
  .

Exemplos das áreas visíveis:

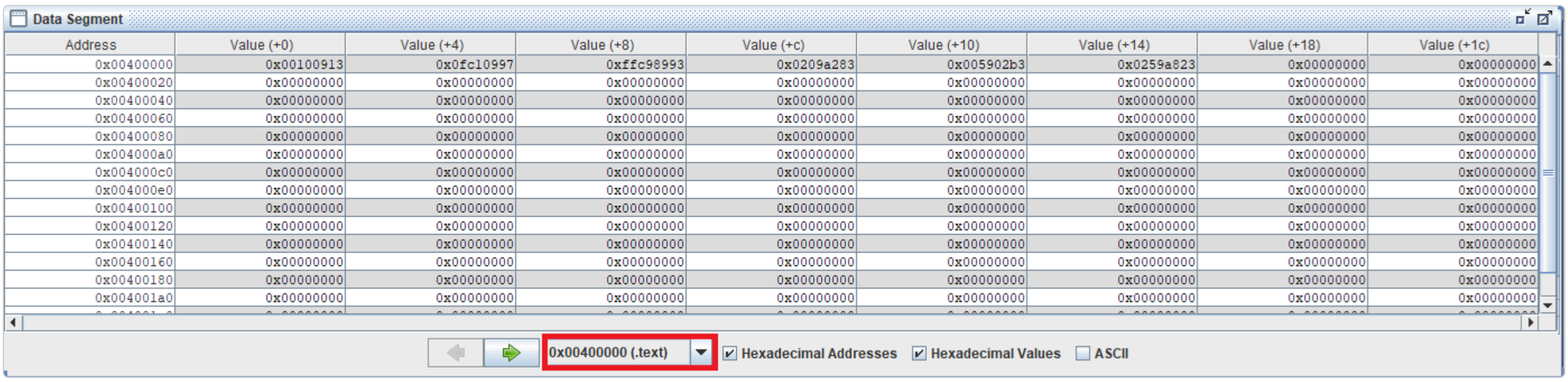
Área do segmento de dados



Área da pilha



Área do segmento de texto



Na seção DATA, os elementos do vetor são armazenados em células de memória organizadas de forma matricial, sendo que o endereço inicial do vetor é igual a **0x10010000** conforme a tabela a seguir:

Endereço da linha	Deslocamento							
	(+0)	(+4)	(+8)	(+12)	(+16)	(+20)	(+24)	(+28)
0x10010000	**Array_A[0]**	**Array_A[1]**	**Array_A[2]**	**Array_A[3]**	**Array_A[4]**	**Array_A[5]**	**Array_A[6]**	**Array_A[7]**
0x10010020	**Array_A[8]**	**Array_A[9]**	**Array_A[10]**	**Array_A[11]**	**Array_A[12]**	**Array_A[13]**	**Array_A[14]**	**Array_A[15]**

Entendendo os endereços dos elementos

Cada elemento do vetor ocupa 4 bytes na memória. Para saber o endereço de um elemento, somamos o endereço da linha com o deslocamento da coluna.

Por exemplo, para localizar o elemento **Array_A[11]**, veja os passos:

- Sabemos que ele está na segunda linha, coluna de deslocamento +12.
- O endereço da linha é

  ```
  0x10010020
  ```

  .
- O deslocamento +12 equivale a

  ```
  0xC
  ```

  em hexadecimal.
- Somando:

  ```
  0x10010020 + 0xC = 0x1001002C
  ```

Outra forma de calcular, a partir do endereço base do vetor:

```
0x10010000 + (4 x 11) = 0x10010000 + 44 = 0x10010000 + 0x2C =
      0x1001002C
```

Portanto, usamos:

```
(end. base em hexa) + (4 x posArray dec.) = (end. desejado)
```

Importante

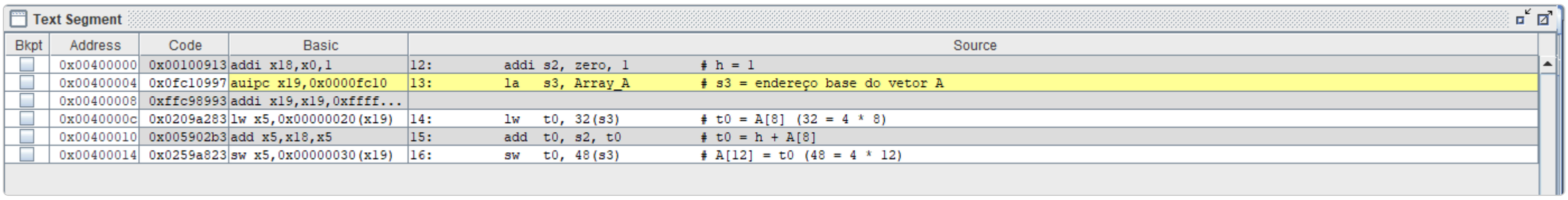
Os valores definidos no vetor **Array_A** são escritos em decimal no código-fonte, mas armazenados na memória em formato hexadecimal. Por exemplo:

- **Array_A[8]** no código é **80** (decimal).
- Na memória, aparece como **0x00000050**.

De acordo com o código, o registrador **s2** guarda o valor **1** e será somado ao conteúdo de **Array_A[8]**. O resultado será armazenado em **Array_A[12]**.

Execute o programa passo-a-passo clicando no botão **Run One Step** até chegar à segunda instrução (endereço **0x00400004**)

Abaixo, observe, na terceira coluna, que a instrução a ser executada é **auipc x19 64528** (referente a **Array_A**), mas que originalmente foi especificada como **la s3, Array_A**. A instrução **la** (load address) é uma pseudo-instrução que o montador traduz para uma sequência das instruções reais **auipc** (Add Upper Immediate to PC) e **addi** no RISC-V. Essa sequência é responsável por carregar o endereço-base do vetor (**Array_A**) no registrador **s3**, possibilitando o acesso aos seus elementos por meio de deslocamentos.



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			**Depois da execução da instrução**									
PC	**Instrução**	**Pseudoinstrução**	**Registradores**			**Segmento de Dados**						
			R5	**R18**	**R19**	**0x10010000**	**...**	**0x10010020**	**...**	**0x10010030**	**...**
			(t0)	**(s2)**	**(s3)**	**Array_A[0] 1º linha coluna(+0)**	**...**	**Array_A[8] 2º linha coluna(+0)**	**...**	**Array_A[12] 2º linha coluna(+10)**	**...**
			0x00000000	**0x00000000**	**0x00000000**	**0x00000000**		**0x00000050**		**0x00000078**	
0x00400000	**addi s2, zero, 1**			**0x00000001**							
0x00400004	**auipc x19 64528**	**la s3, Array_A**			**0x10010004**						
0x00400008	**addi x19 x19 -4**				**0x10010000**						
0x0040000c	lw x5, 0x004000	lw t0, 32(s3)	0x00000050	0x00000001	0x10010000	0x00000000		0x00000050		0x00000078	
0x00400010	add x5, x18, x5	add t0, s2, t0	0x00000051	0x00000001	0x10010000	0x00000000		0x00000050		0x00000078	
0x00400014	sw x5, 0x000000	sw t0, 48(s3)	0x00000051	0x00000001	0x10010000	0x00000000		0x00000050		0x00000051	

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

Salvar como PDF

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

← Voltar ao tutorial