|  |
| --- |
| Exercício 02 |
| **Objetivo:**  Aprender a utilizar a interface do RARS executando o segundo exemplo de programação na linguagem de montagem do RISC-V no livro texto. |
| **Instruções:**   1. Inicie o RARS 2. No editor de textos do RARS, transcreva o código abaixo e salve o arquivo com o nome **exercicio\_02**.   ####################################################################  # Exercício 02 - Patterson pags. 54/55/56  # Mostra a compilação de um comando de atribuição em C usando Array  ####################################################################  # Trecho em C:  #  # A[12] = h + A[8]  .data # segmento de dados  # definição do array A. Coloca os valores de A[0]=0 até A[15]=150 na memória  Array\_A: .word 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150  .text # segmento de código (programa)  main:  addi s2, zero, 1 # Inicializa s2 em 1  la s3, Array\_A # como o exercício assume que o endereço-base de A[]  # está em s3, foi incluida esta instrução  lw t0, 32(s3) # t0 = A[8]  add t0, s2, t0 # t0 = t0 + h  sw t0, 48(s3) # A[12] = t0 |
| 1. Para iniciar a montagem do código vá ao menu **Run** e selecione a opção **Assemble** ou pressione **F3**. 2. Na janela de segmento de dados se pode optar por mostrar, por exemplo, o conteúdo da região de memória que armazena os dados do programa (0x10010000 (.data)), da pilha (current sp) e do sistema operacional (0xffff0000 (.MMIO)), conforme é mostrado abaixo.   **Área de dados do sistema operacional**    **Área de dados do programa do usuário**    **Área de dados da pilha** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. 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) | (+c) | (+10) | (+14) | (+18) | (+1C) | | 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] | | 0x10010040 |  |  |  |  |  |  |  |  | | 0x10010060 |  |  |  |  |  |  |  |  |   O endereço de cada elemento na matriz é dado pelo endereço da linha somado ao deslocamento associado à coluna. Por exemplo, o endereço do elemento 11 é dado por 0x10010020 + 0xC = 0x1001002C.  Lembre que, se o cálculo for feito em relação ao endereço base do vetor, deve-se fazer: 0x10010000 + 4 x 11 = 0x10010000 + 44 = 0x10010000 + 0x2C = 0x1001002C.  (end. base em hexa)+(4 x posArray dec.) = (end. base em hexa) + ( desl. decimal) = **(end. base hexa) + (desl. hexa)** = (end. desejado).   1. Observe que os valores armazenados no vetor são expressos em hexadecimal, embora no código eles sejam definidos em decimal. Então, por exemplo, o elemento Array\_A[8] = 0x00000050 = 80 (em decimal). 2. Conforme especificado no código do programa, o que se espera, após a sua execução, é que o elemento Array\_A[12] receba a soma do conteúdo do elemento Array\_A[8] com o conteúdo do registrador s2. Se este registrador for igual a 0, então, após a execução, Array\_A[12] será igual a Array\_A[8]. 3. Inicie a execução passo-a-passo, pressionando **F7** até chegar à segunda instrução do programa (endereço 0x00400004). 4. Abaixo, observe, na quarta coluna, que a instrução a ser executada é **auipc 19, 0xfc10 [Array\_A]**, mas que originalmente foi especificada como **la s3, Array\_A**. A instrução **la** (load array) é na verdade uma pseudo-instrução que o montador traduz para uma seqüência das instruções **auipc** (add upper immediate to pc) e **addi** (add immediate) do RISC-V (pseudo-instruções tornam a programação mais facilitada). Essa sequência permite carregar o endereço-base do vetor (Array\_A) para um registrador base (s3) para que se possa acessar qualquer elemento do vetor por meio de deslocamentos em relação ao endereço-base. |
| 1. 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 da execução da instrução** | | Depois da execução da instrução | | | | | | | | |
| **Registradores** | | | **Segmento de Dados** | | | | | |
| **PC** | **Instrução** | **R8** | **R18** | **R19** | **10010000** | **…** | **10010040** | **…** | **10010060** | **…** |
| **($t0)** | **($s2)** | **($s3)** | Array\_A[0] 1a linha  coluna (+0) | … | Array\_A[8] 3a linha  coluna (+0) | … | Array\_A[12] 4a linha  coluna (+0) | … |
|  |  | 00000000 | 00000000 | 00000000 | 00000000 |  | 00000050 |  | 00000078 |  |
| **00400000** | **addi s2, zero, 1** |  | **00000001** |  |  |  |  |  |  |  |
| 00400004 | **auipc 19, Array\_A** |  |  |  |  |  |  |  |  |  |
| 00400008 | **addi 19,19, 0xfffffffc** |  |  | **1001000** |  |  |  |  |  |  |
| 0040000C | **lw x5, 0x00000020(x19)** | **00000050** |  |  |  |  |  |  |  |  |
| 00400010 | **Add x5, x5, x18** |  | 00000051 |  |  |  |  |  |  |  |
|  | **sw x5 0x00000030(x19)** |  |  |  |  |  |  |  |  |  |

**Obs**: A segunda (**auipc**) e a terceira (**addi**) instruções apresentadas acima são inseridas pelo montador em substituição à pseudo-instrução **la s3, Array\_A** utilizada originalmente no código fonte. Essa pseudo-instrução tem a função de carregar o endereço base do vetor (**Array\_A**) para o registrador destino (**s3**). A instrução **auipc** primeiramente carrega o endereço do vetor para o registrador **19** e a segunda instrução soma o conteúdo de **19** para **s3**.

|  |
| --- |
| NOTA: Se for necessário reiniciar o programa, faça: **Run > Reset** |