| **Exercício 02** |
| --- |
| **Objetivo:**  Aprender a utilizar a interface do MARS executando o segundo exemplo de programação na linguagem de montagem do MIPS no livro texto. |
| **Instruções:**   1. Inicie o MARS 2. No editor de textos do MARS, 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 (0x90000000 (.kdata)), 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] |   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 **lui $19, 4097 [Array\_A]**, mas que originalmente foi especificada como **la $s3, Array\_A**. A instrução **la** (load array) é na verdade uma pseudo-instrução que o montador traduz para uma seqüência das instruções **lui** (load upper immediate) e **ori** (or immediate) do MIPS (pseudo-instruções tornam a programação mais facilitada). Essa seqüência permite carregar o endereço-base do vetor (Array\_A) para um registrador base ($s2) 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** | **…** | **10010020** | **…** | **10010030** | **…** |
| **($t0)** | **($s2)** | **($s3)** | **Array\_A[0]**  1a linha  coluna (+0) | **…** | **Array\_A[8]**  2a linha  coluna (+0) | **…** | **Array\_A[12]**  2a linha  coluna (+10) | **…** |
|  |  | 00000000 | 00000000 | 00000000 | 00000000 |  | 00000050 |  | 00000078 |  |
| **00400000** | **addi $s2, $zero, 1** |  | **00000001** |  |  |  |  |  |  |  |
| 00400004 | **lui $at, Array\_A** |  |  | **1001000** |  |  |  |  |  |  |
| 00400008 | **ori $s3, $at, 0** |  |  | **1001000** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

**Obs**: A segunda (**lui**) e a terceira (**ori**) 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 **lui** primeiramente carrega o endereço do vetor para o registrador **$at** (não listado acima) e a segunda instrução transfere o conteúdo de **$at** para **$s3**.

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