ATIVIDADE AVALIATIVA

Instruções - 2

Estudante: Wagner Clemente Coelho Batalha 20/0044486

Estudante: Eder de Amaral Amorim 17/0140636

Obs.: a) Cole no texto de solução de cada exercício *print* de tela do simulador MARS (ou DrMIPS) com o código em assembly, tabela de registradores (abra Registers) e tabela de memória RAM (Data Segment). Use-o para buscar erros de sintaxe.

b) Introduza valores nos registradores e memória (manualmente, diretamente nas tabelas) para verificar o funcionamento do código assembly. Por padrão, há somente zero em todas as posições no início da execução.

c) Para carregar (“atribuir”) valor a um registrador via código assembly, é possível usar a instrução addi. Ex.: addi $t0, $zero, 2 (o efeito é $t0=2). Isso ajuda a verificar o funcionamento do código.

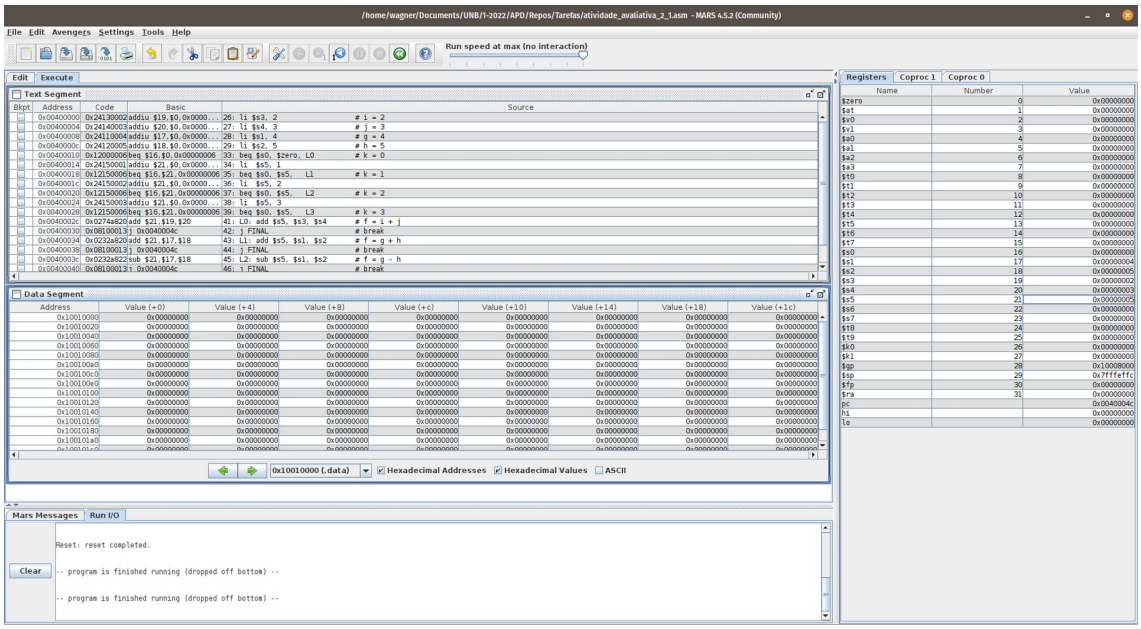
d) Caso não for possível visualizar o valor na RAM (Data Segment), altere o offset (load, store) para um número negativo (-10, -100, -1000, -10000 etc) até ele constar na tela. Escreva um comentário (#) no código assembly para avisar que foi necessário ajustar o offset. Essa problemática de *range* de endereços de memória depende da versão do MARS; basta avisar esse problema em comentário no código assembly que o/a estudante não perderá pontos na correção.

Escreva em *assembly* do MIPS o seguinte código em C.

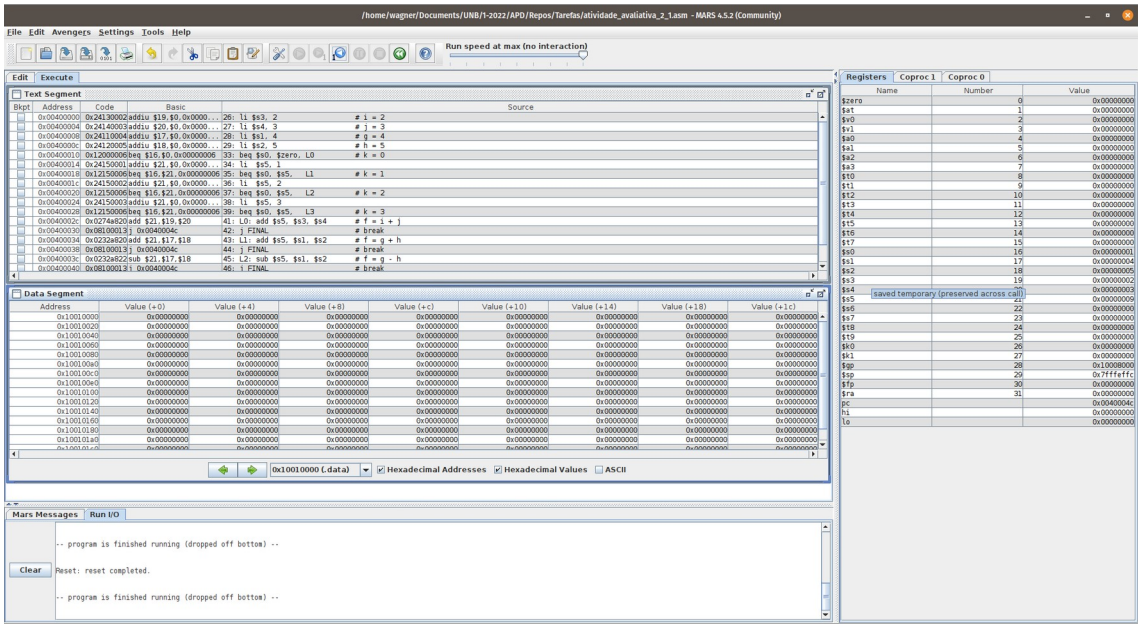
Sugestões: comece reescrevendo esse código como uma sequência de quatro *if-then*; no assembly, o *break* pode ser implementado como um *jump* (j) para a última linha do código (sai dessa estrutura); atenção para pular para o final do código após executar o equivalente de um *case*.

```
switch(k)
{
  case 0:f=i+j; break;
  case 1:f=g+h; break;
  case 2:f=g-h; break;
  case 3:f=i-j; break;
}
```

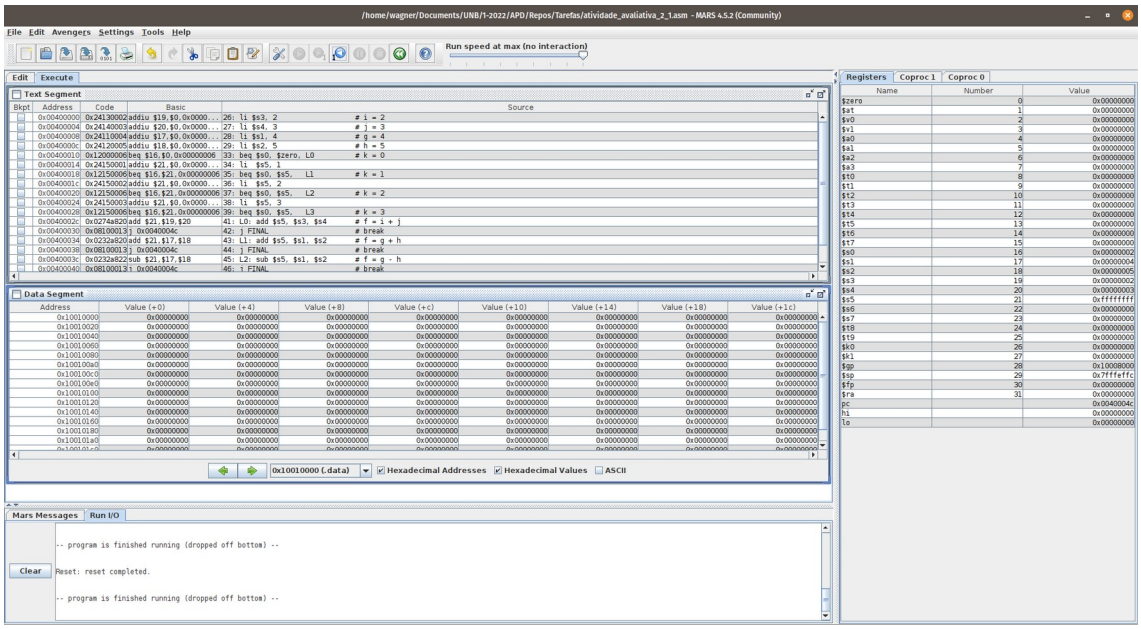
```
# if(k==0){
#     f=i+j;
#     goto FINAL;
# } else if (k==1){
#     f=g+h;
#     goto FINAL;
# } else if (k==2){
#     f=g-h;
#     goto FINAL;
# } else if (k==3){
#     f=i-j;
#     goto FINAL;
# }
#
# FINAL:

# k = $s0
# g = $s1
# h = $s2
# i = $s3
# j = $s4
# f = $s5

#########################
# Valores de debug
li $s3, 2                           # i = 2
li $s4, 3                           # j = 3
li $s1, 4                           # g = 4
li $s2, 5                           # h = 5
#########################


beq $s0, $zero, L0              # k = 0
li  $s5, 1
beq $s0, $s5,   L1              # k = 1
li  $s5, 2
beq $s0, $s5,   L2              # k = 2
li  $s5, 3
beq $s0, $s5,   L3              # k = 3

L0: add $s5, $s3, $s4           # f = i + j
j FINAL                             # break
L1: add $s5, $s1, $s2           # f = g + h
j FINAL                             # break
L2: sub $s5, $s1, $s2           # f = g - h
j FINAL                             # break
L3: sub $s5, $s3, $s4           # f = i - j
j FINAL
FINAL:
```



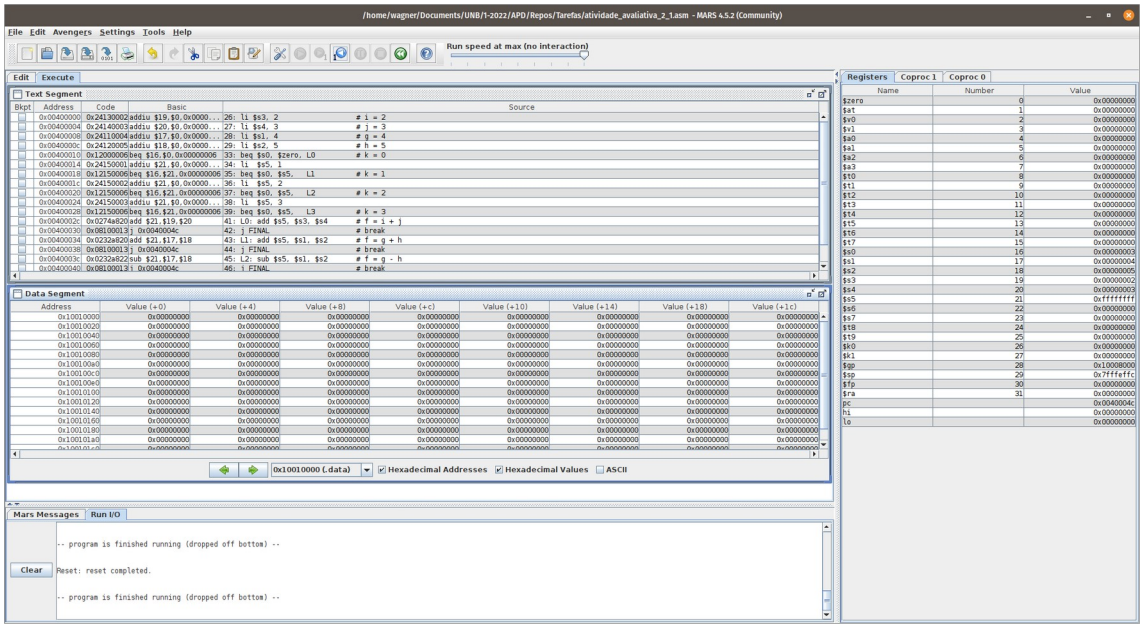
Quando k = 0



Quando k = 1



Quando K = 2



Quando K = 3