```
# EXERCICIO 6 - IMPLEMENTAÇÃO STRCMP

                .data
                .align 0
str_welcome:          .asciiz ":: Implementacao da funcao strcmp ::\n>
int strcmp(const char *str1, const char *str2) <\n\n"
str_str1:       .asciiz " Entre com a string 1 para a funcao: "
str_str2:       .asciiz " Entre com a string 2 para a funcao: "
str_result:     .asciiz "\n Resultado: "

                .align 2
str1_sz:        .word 50
str2_sz:        .word 50

                .text

                .globl main
main: # 1. Imprimindo as boas vindas :) ------------------------
      la    $a0,  str_welcome      # load str_welcome
      li    $v0,  4                # print str code
      syscall                      # system, do it!

      # 2. Alocando espaco para a string1 e para a string2 -----
      lw    $a0,  str1_sz          # load string1 size
      li    $v0,  9                # alloc memory code
      syscall                      # system, do it!
      move  $s0,  $v0              # $s0 = &str1

      lw    $a0,  str2_sz          # load string2 size
      li    $v0,  9                # alloc memory code
      syscall                      # system, do it!
      move  $s1,  $v0              # $s1 = &str2

      # 3. Lendo a string1 -------------------------------------
      la    $a0,  str_str1         # load str_str1
      li    $v0,  4                # print str code
      syscall                      # system, do it!

      move  $a0,  $s0              # $a0 = &str1
      lw    $a1,  str1_sz          # $a1 = str1 size
      li    $v0,  8                # read string code
      syscall

      # 4. Lendo a string2 -------------------------------------
      la    $a0,  str_str2         # load str_str2
      li    $v0,  4                # print str code
      syscall                      # system, do it!

      move  $a0,  $s1              # $a0 = &str2
      lw    $a1,  str2_sz          # $a1 = str2 size
      li    $v0,  8                # read string code
      syscall

      # 5. Realizando a comparacao -----------------------------
      move  $a0,  $s0              # $a0 = &str1
      move  $a1,  $s1              # $a1 = &str2
      jal   strcmp                 # do strcmp
      add   $s2,  $zero,     $v0   # $s2 = result of strcmp
```

```
      # 6. Imprimindo na tela o resultado
      la    $a0,  str_result       # load str_result
      li    $v0,  4                # print str code
      syscall                      # system, do it!

      move  $a0,  $s2              # load result
      li    $v0,  1                # print int code
      syscall                      # system, do it!

      # 7. Encerrando o programa
      li    $v0,  10               # exit code
      syscall                      # system, do it!

#:::::::::::::::::: STRCMP ::::::::::::::::::::
strcmp:
      # 1. Ajustando a pilha para a funcao
      add   $sp,  $sp,  -12        # stack build
      sw    $a0,  8($sp)           # save &str1 in the stack
      sw    $a1,  4($sp)           # save &str2 in the stack
      sw    $ra,  0($sp)           # save return address in the stack

      # 2. Comparando a string1 com a string2
      lw    $t6,  str1_sz          # $t6 = str1_sz
      lw    $t7,  str2_sz          # $t7 = str2_sz

      # a) se o tamanho for diferente, encerra-se a funcao sem comprar
byte a byte...
      sub   $t5,  $t6,  $t7              # $t5 = $t6 - $t7
      bne   $t5,  $zero,     end_strcmp # if $t5 != 0, goto end_strcmp

      # b) caso o tamanho seja o mesmo, realiza-se a comparacao byte a
byte...
      add   $t6,  $t6,  $a0        # $t6 = &str1 + str1_sz
      add   $t7,  $t7,  $a1        # $t7 = &str2 + str2_sz
cpy_loop:
      lb    $t0,  0($a0)           # $t0 = str1[i]
      lb    $t1,  0($a1)           # $t1 = str2[i]

      sub   $t5,  $t0,  $t1   # $t5 = $t6 - $t7
      bne   $t5,  $zero,     end_strcmp # if $t5 != 0, goto end_strcmp

      add   $a0,  $a0,  1          # $a0 += 1
      add   $a1,  $a1,  1          # $a1 += 1
      bne   $t0,  $zero,     cpy_loop   # if $a0 != '\0' goto cpy_loop

end_strcmp:
      # 3. Atribuindo a resposta a $v0
      add   $v0,  $zero,     $t5   # $v0 = $t5

      # 4. Reajustando a pilha para retornar para a chamada da funcao
      lw    $a0,  8($sp)           # load &str1 in the stack
      lw    $a1,  4($sp)           # load &str2 in the stack
      lw    $ra,  0($sp)           # load return address in the stack
      add   $sp,  $sp,  -12        # stack free

      jr    $ra                    # return to call_func address
```