```
# EXERCICIO 3 - IMPLEMENTAÇÃO NUMINVERTER
                .data
                .align 0
dig_txt: .asciiz "Digite um numero inteiro valido (entre 0 a 999): "
inv_txt: .asciiz "O inverso eh: "
                .text
                .globl main
main:
      # 1. Lendo o numero de entrada que e 0 <= num <= 999 -------------

read:
      li    $v0,  4                 # print string code
      la    $a0,  dig_txt           # load dig_txt
      syscall                       # system, do it!

      li    $v0,  5                 # read int code
      syscall                       # system, do it!
      add   $t0,  $zero,      $v0   # $t0 = input num

      blt   $t0,  0,    read        # if num < 0, read another
      bgt   $t0,  999,  read        # if num > 999, read another

      move  $s0,  $t0               # salvando o valor de entrada

      # 2. Iniciando a funcao de inverter os dígitos --------------------
      jal numinverter               # do numinverter
      move  $s1,  $v0               # salvando o resultado da funcao

      # 3. Imprimindo o resultado ---------------------------------------
      li    $v0,  4                 # print string code
      la    $a0,  inv_txt           # load inv_txt
      syscall                       # system, do it!

      li    $v0,  1                 # print int code
      move  $a0,  $s1               # load result
      syscall                       # system, do it!

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

#:::::::::::::::::: NUMINVERTER ::::::::::::::::::::
numinverter:
      # 1. Criando a pilha da funcao
      addi  $sp,  $sp,  -8
      sw    $s0,  4($sp)            # salvando o numero entrado (ex.: 128)
      sw    $ra,  0($sp)            # salvando o endereco de retorno

      # 2. Descobrindo os dígitos do numero
      # (a) dig mais sig
      div   $v0,  $s0,  100         # v0 = dig mais sign(ex: 128 / 100 = 1)
      move  $s1,  $v0               # s1 = v0 (1)

      # (b) dig do meio
      mul   $t0,  $s1,  100         # t0 = dig mais sign*100 (t0 = 1 * 100)
      sub   $v0,  $s0,  $t0         # v0 = numero entrado - dig mais sign
      div   $v0,  $v0,  10          # v0 = dig do meio (v0 = 28 / 10 = 2)
      move  $s2,  $v0               # s2 = v0 (2)
```

```
      # (c) dig menos sig
      mul   $t0,  $s1,  100         # t0 = dig mais sign*100 (t0 = 1 * 100)
      sub   $v0,  $s0,  $t0         # v0 = numero entrado-dig mais sign*100
      mul   $t0,  $s2,  10          # t0 = digito do meio*10 (t0 = 2 * 10)
      sub   $v0,  $v0,  $t0         # v0 = v0 - dig do meio* 10
      move  $s3,  $v0               # s3 = v0 (8)

      # 3. Formando o novo numero
      li    $v0,  0                 # result = 0
      bge   $s1,  1,    if3d        # se ele tiver 3 digitos
      bge   $s2,  1,    if2d        # se ele tiver 2 digitos
      bge   $s3,  1,    if1d        # se ele tiver 1 digito
if3d:
      mul   $t0,  $s3,  100         # $t0 = dig menos sig * 100
      add   $v0,  $v0,  $t0         # result += $t0
      mul   $t0,  $s2,  10          # $t0 = dig do meio * 10
      add   $v0,  $v0,  $t0         # result += $t0
      add   $v0,  $v0,  $s1         # result += dig mais sig

      j endifs
if2d:
      mul   $t0,  $s3,  10          # $t0 = dig menos sig * 10
      add   $v0,  $v0,  $t0         # result += $t0
      add   $v0,  $v0,  $s2         # result += digito mais sig
      j endifs
if1d:
      add   $v0,  $v0,  $s3         # result = unico digito

endifs:

      # 5.Liberando a pilha e encerrando a funcao
      lw    $s0,  4($sp)            # load num
      lw    $ra,  0($sp)            # load ra
      sub   $sp,  $sp,  -8          # readjust stack

      jr $ra                        # return to func call
```