Biblioteca “util.s”

`void print_rax_as_string();`
Exibe uma string na tela onde o ponteiro para essa string está em %rax.

`void print_rax_as_long();`
Exibe o conteúdo de %rax na tela como um inteiro 64-bits com sinal.

`void print_rax_as_ulong();`
Exite o conteúdo de %rax na tela como um inteiro 64-bits sem sinal.

`void print_rax_as_hex();`
Exibe o conteúdo de %rax na tela na foma de hexadecimal.

`void print_eax_as_int();`
Exite o conteúdo de %eax na tela como um inteiro de 32-bits com sinal.

`void print_eax_as_uint();`
Exite o conteúdo de %eax na tela como um inteiro de 32-bits com sinal.

`void read_int_to_eax();`
Lê um inteiro (32-bits com sinal) do teclado e coloca o valor em %eax.

`void read_long_to_rax();`
Lê um inteiro (64-bits com sinal) do teclado e coloca o valor em %rax.

`void read_string_to_rax();`
Lê uma string do teclado e armazena em um buffer.
O ponteiro do buffer deve estar armazenado em %rax e a função assume que o tamanho do buffer é fixo com 128 bytes.

Exemplo 01

O código abaixo é de um programa “test-01.s” que chama as funções (usando *call*) da biblioteca “util.s”. Para compilar o programa use a seguinte linha:

```
gcc  -o  test-01 test-01.s  util.s
```

test-01.s

```
.data

str01:  .string "Digite o valor de x:"
str02:  .string "O valor de x e:"

.text

.globl main
main:
  #-- Inicializacao: não mexa ----------------------------------------
  pushq   %rbp
  movq    %rsp, %rbp
  subq    $16, %rsp
  #-------------------------------------------------------------------

  movq $str01, %rax                 # rax = &str01
  call print_rax_as_string          # exibe a string na tela

  call read_int_to_eax              # le do teclado e salva em eax
  movl %eax, %ecx                   # salva o valor em ecx

  movq $str02, %rax                 # rax = &str02
  call print_rax_as_string          # exibe a string na tela

  movl %ecx, %eax                   # retorna o valor para eax
  call print_eax_as_int             # mostra o numero na tela

  #-- Finalizacao: não mexa ------------------------------------------
  movl $0, %eax
  leave
  ret
```

Exemplo 02

O código abaixo é de um programa “test-02.s” que chama as funções (usando *call*) da biblioteca “util.s”. Para compilar o programa use a seguinte linha:

```
gcc  -o  test-02 test-02.s  util.s
```

test-02.s

```
.data

str01: .string "Digite uma frase:"
str02: .string "Voce digitou:"

buf: .zero 128    # array de 128 bytes com zero

.text

.globl main
main:
  #-- Inicializacao: não mexa ----------------------------------------
  pushq   %rbp
  movq    %rsp, %rbp
  subq    $16, %rsp
  #-------------------------------------------------------------------

  movq $str01, %rax               # rax = &str01
  call print_rax_as_string        # mostra 'str01' na tela

  movq $buf, %rax                 # rax = &buf
  call read_string_to_rax         # le string para o buffer

  movq $str02, %rax               # rax = &str02
  call print_rax_as_string        # mostra 'str02' na tela

  movq $buf, %rax                 # rax = &buf
  call print_rax_as_string        # mostra 'buf' na tela

  #-- Finalizacao: não mexa ------------------------------------------
  movl $0, %eax
  leave
  ret
```