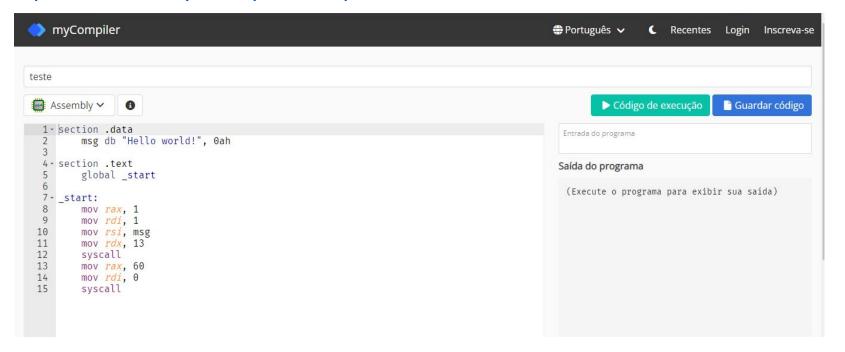
Simulador

https://www.mycompiler.io/pt/new/asm-x86 64?fork=Kn6PYf3VBmr



Integer value	Name	<pre><unistd.h> symbolic constant^[1]</unistd.h></pre>	<stdio.h> file stream[2]</stdio.h>
0	Standard input	STDIN_FILENO	stdin
1	Standard output	STDOUT_FILENO	stdout
2	Standard error	STDERR_FILENO	stderr

https://blog.rchapman.org/posts/Linux System Call Table for x8 6 64/

Registradores

- RAX Um registrador "acumulador", usado em instruções aritméticas.
- RBX Registrador base, usado para endereçamento de base.
- RCX Usado em instruções de repetição loop.
- RDX Armazena dados durante operações entrada/saída.
- RDI Índice de destino em comandos de manipulação de string.
- RSI Índice de origem em comandos de manipulação de string.

RBP-Base do stack frame

RSP - Armazena endereço do topo da pilha

R8-R15 - Utilizados para armazenar variáveis temporárias

string. dados

Movimentação de

só devem ser usados preferencialmente para trabalhar com pilha. Aloque o valor 30 para o registrador RAX e o valor 45 para o registrador RBX, some.

Por fim, utilize a syscall para imprimir.

Qual letra foi impressa?

section .bss letra: resb 1

section .text global _start

_start:

mov rax, 30 mov rbx, 45 add rax, rbx mov [letra], rax

mov rax, 1 mov rdi, 1 mov rsi, letra mov rdx, 1 syscall

mov rax, 60 mov rdi, 0 syscall Faça um programa que receba 1 letra minúscula e converta para maiúscula.

```
section .bss
letra: resb 1
letra_mai: resb 1
section text
  global _start
_start:
  mov rax, 0
  mov rdi, 0
  mov rsi, letra
  mov rdx, 1
  syscall
  mov rbx, [letra]
  sub rbx, 32
  mov [letra_mai], rbx
  mov rax, 1
  mov rdi, 1
  mov rsi, letra_mai
  mov rdx, 1
  syscall
  mov rax, 60
  mov rdi. 0
  syscall
```

Faça um programa que calcule o tamanho de uma string.

```
section .data
msg: db "hello",0
section .text
strlen:
  mov rax, 0
  .loop:
     cmp byte [rdi+rax],0
     je .end
     inc rax
     jmp .loop
  .end:
     ret
global _start
_start:
     mov rdi, msg
     call strlen
     mov rdi, rax
     mov rax, 60
     syscall
```

```
section .data
   msg: db "hello",0
   section .text
 7 - strlen:
 8
        mov rax, 0
10 -
        .loop:
11
            cmp byte [rdi+rax],0
12
            je .end
13
            inc rax
14
            jmp .loop
15 -
        .end:
16
            ret
17 global _start
18 - _start:
19
            mov rdi, msg
20
            call strlen
21
            mov rdi, rax
22
23
            mov rax, 60
24
            syscall
25
26
```