

# MIPS UTILIZANDO MARS

Aula.05

**Curso Tecnologia em Análise e Desenvolvimento de  
Sistemas**

Prof. Mauro

26/11/2024



# MARS?

- No MIPS, MARS (MIPS Assembler and Runtime Simulator) é um simulador de código Assembly utilizado para ensinar e aprender a programar em linguagem Assembly para a arquitetura MIPS.
  - Ele permite que os usuários escrevam, simulem e depurem programas MIPS sem a necessidade de um hardware físico, proporcionando uma interface gráfica para interação.

# CARACTERÍSTICAS DO MARS

1. Simulação de Código Assembly MIPS: MARS permite escrita e execução de programas em Assembly para a arquitetura MIPS, que é uma arquitetura RISC (Reduced Instruction Set Computer).
2. Depuração: Ele oferece funcionalidades de depuração, como execução passo a passo, visualização dos registradores, memória, e outras informações de execução, para que você possa entender o comportamento do seu código.
3. Visualização da Memória e Registradores: É possível observar como a memória e os registradores da CPU são manipulados à medida que seu programa MIPS é executado, o que é útil para aprender sobre como o hardware funciona.

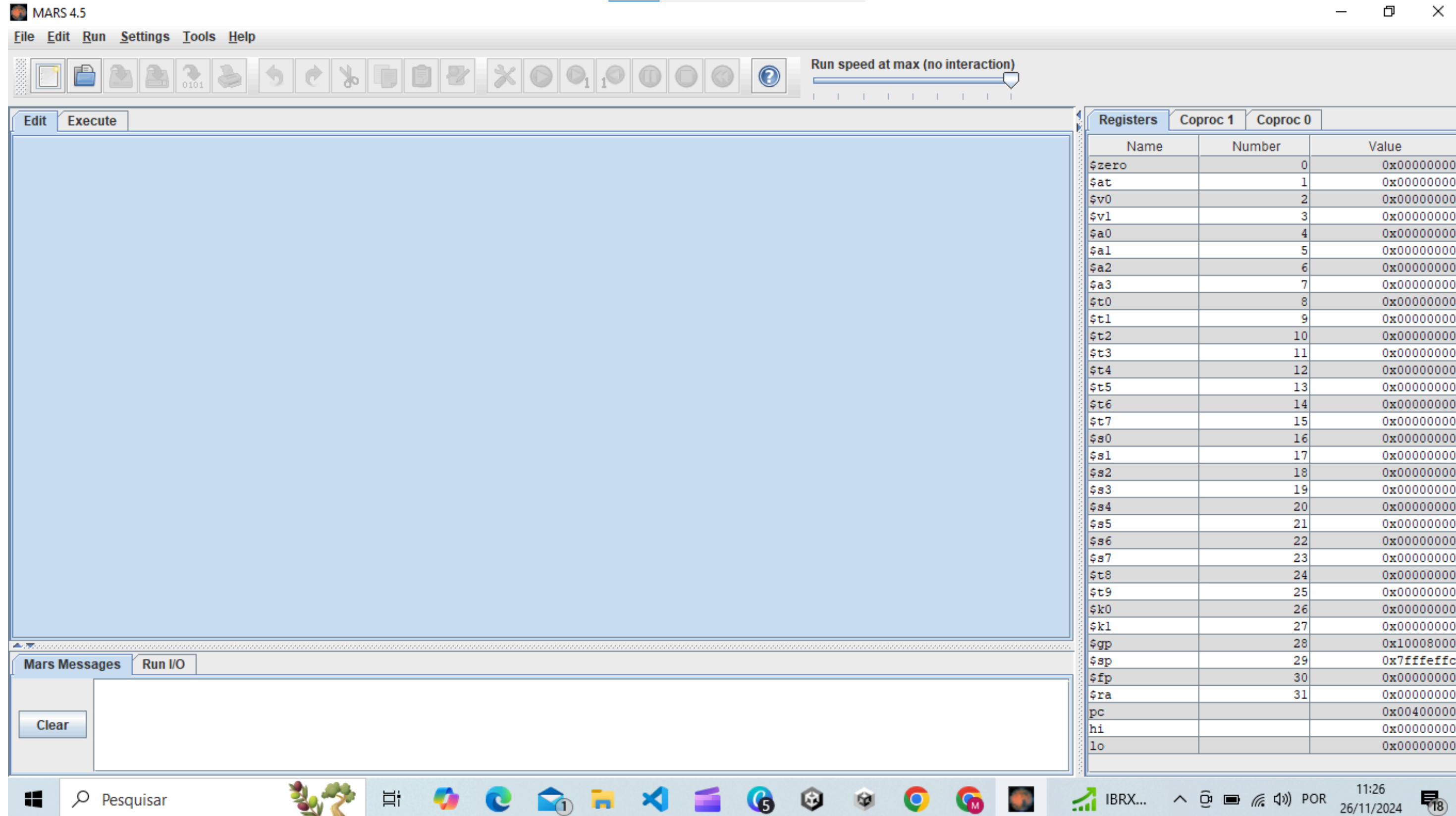
# CARACTERÍSTICAS DO MARS

4. Interface Gráfica de Fácil Uso: MARS tem uma interface amigável que facilita o processo de escrever e simular programas MIPS, o que o torna uma excelente ferramenta para iniciantes em arquitetura de computadores e programação de baixo nível.

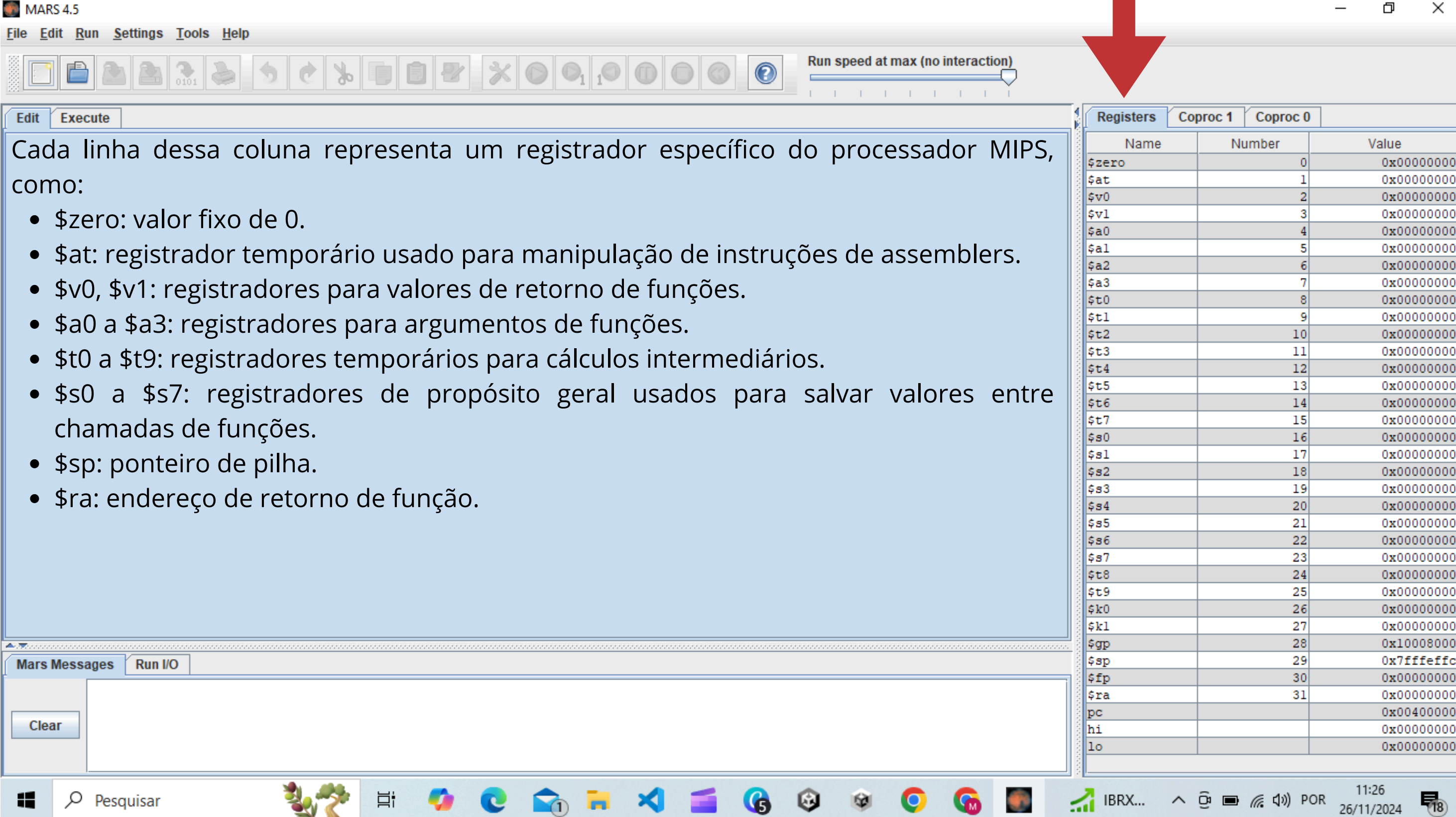
5. Execução de Programas: O MARS simula a execução de um programa MIPS e fornece informações detalhadas sobre o fluxo de controle, como a execução de instruções, o conteúdo dos registradores e o estado da memória.

6. Facilidade de Uso para Educação: Ele é amplamente utilizado em cursos de arquitetura de computadores e sistemas embarcados, onde os alunos aprendem conceitos fundamentais como gerenciamento de memória, manipulação de registradores e execução de instruções de baixo nível.

# MARS - Interface



# MARS - Interface



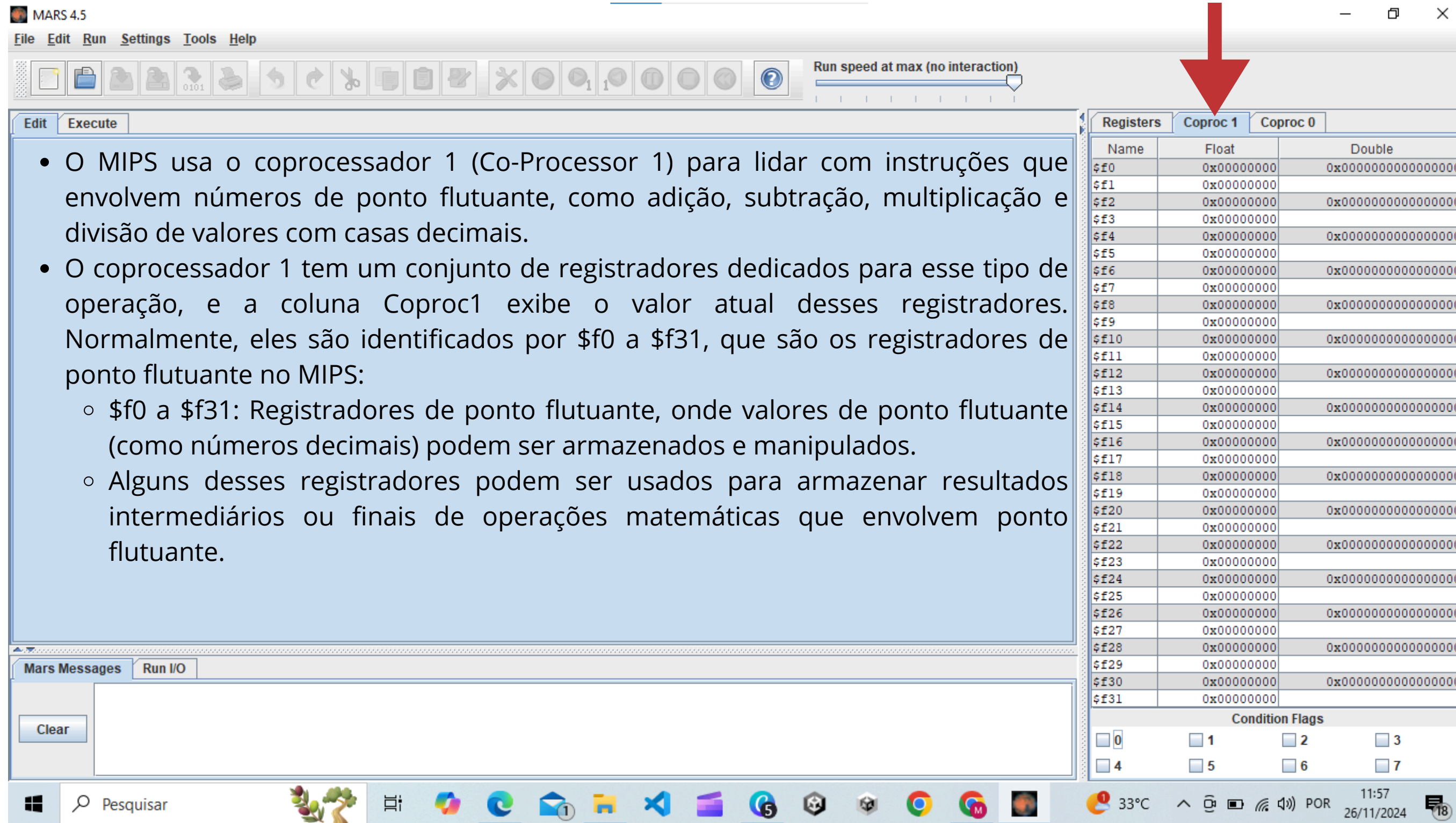
Cada linha dessa coluna representa um registrador específico do processador MIPS, como:

- \$zero: valor fixo de 0.
- \$at: registrador temporário usado para manipulação de instruções de assemblers.
- \$v0, \$v1: registradores para valores de retorno de funções.
- \$a0 a \$a3: registradores para argumentos de funções.
- \$t0 a \$t9: registradores temporários para cálculos intermediários.
- \$s0 a \$s7: registradores de propósito geral usados para salvar valores entre chamadas de funções.
- \$sp: ponteiro de pilha.
- \$ra: endereço de retorno de função.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000



# MARS - Interface



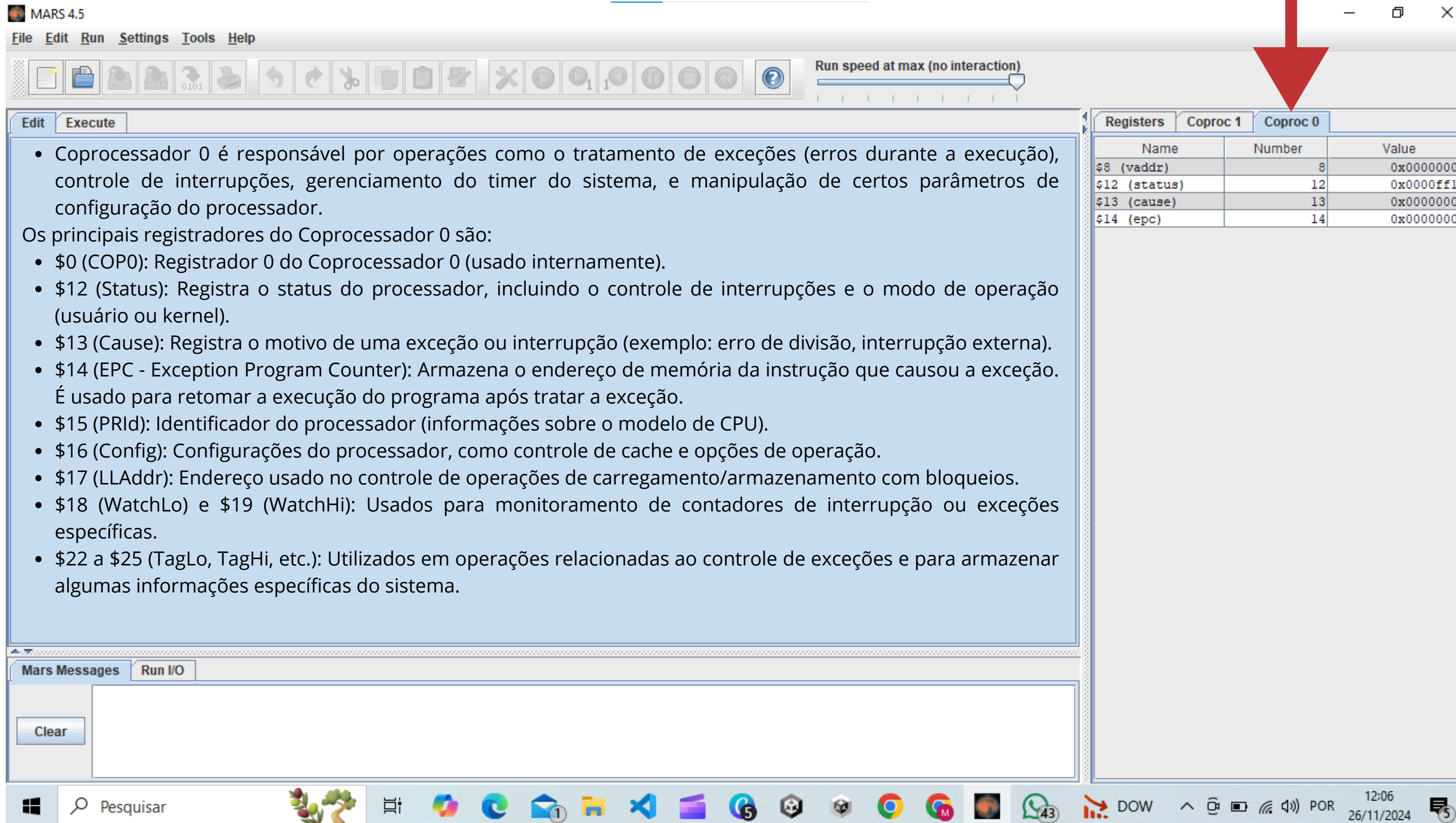
The screenshot shows the MARS 4.5 interface. The main window displays a list of MIPS instructions. The right panel shows the 'Registers' section, with the 'Coproc 1' tab selected. A red arrow points to this tab. The table below shows the values of the Coprocessor 1 registers.

Name	Float	Double
\$f0	0x00000000	0x0000000000000000
\$f1	0x00000000	
\$f2	0x00000000	0x0000000000000000
\$f3	0x00000000	
\$f4	0x00000000	0x0000000000000000
\$f5	0x00000000	
\$f6	0x00000000	0x0000000000000000
\$f7	0x00000000	
\$f8	0x00000000	0x0000000000000000
\$f9	0x00000000	
\$f10	0x00000000	0x0000000000000000
\$f11	0x00000000	
\$f12	0x00000000	0x0000000000000000
\$f13	0x00000000	
\$f14	0x00000000	0x0000000000000000
\$f15	0x00000000	
\$f16	0x00000000	0x0000000000000000
\$f17	0x00000000	
\$f18	0x00000000	0x0000000000000000
\$f19	0x00000000	
\$f20	0x00000000	0x0000000000000000
\$f21	0x00000000	
\$f22	0x00000000	0x0000000000000000
\$f23	0x00000000	
\$f24	0x00000000	0x0000000000000000
\$f25	0x00000000	
\$f26	0x00000000	0x0000000000000000
\$f27	0x00000000	
\$f28	0x00000000	0x0000000000000000
\$f29	0x00000000	
\$f30	0x00000000	0x0000000000000000
\$f31	0x00000000	

Condition Flags

0	1	2	3
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	5	6	7
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

# MARS - Interface



The screenshot displays the MARS 4.5 interface. The main window is divided into several sections. On the left, there is a list of functions for Coprocessor 0. On the right, there is a table showing the current values of the registers. A red arrow points to the 'Coproc 0' tab in the registers section.

**Functions of Coprocessor 0:**

- Coprocessador 0 é responsável por operações como o tratamento de exceções (erros durante a execução), controle de interrupções, gerenciamento do timer do sistema, e manipulação de certos parâmetros de configuração do processador.
- Os principais registradores do Coprocessador 0 são:
- \$0 (COP0): Registrador 0 do Coprocessador 0 (usado internamente).
- \$12 (Status): Registra o status do processador, incluindo o controle de interrupções e o modo de operação (usuário ou kernel).
- \$13 (Cause): Registra o motivo de uma exceção ou interrupção (exemplo: erro de divisão, interrupção externa).
- \$14 (EPC - Exception Program Counter): Armazena o endereço de memória da instrução que causou a exceção. É usado para retomar a execução do programa após tratar a exceção.
- \$15 (PRId): Identificador do processador (informações sobre o modelo de CPU).
- \$16 (Config): Configurações do processador, como controle de cache e opções de operação.
- \$17 (LLAddr): Endereço usado no controle de operações de carregamento/armazenamento com bloqueios.
- \$18 (WatchLo) e \$19 (WatchHi): Usados para monitoramento de contadores de interrupção ou exceções específicas.
- \$22 a \$25 (TagLo, TagHi, etc.): Utilizados em operações relacionadas ao controle de exceções e para armazenar algumas informações específicas do sistema.

**Registers (Coproc 0):**

Name	Number	Value
\$8 (vaddr)	8	0x00000000
\$12 (status)	12	0x0000ff11
\$13 (cause)	13	0x00000000
\$14 (epc)	14	0x00000000



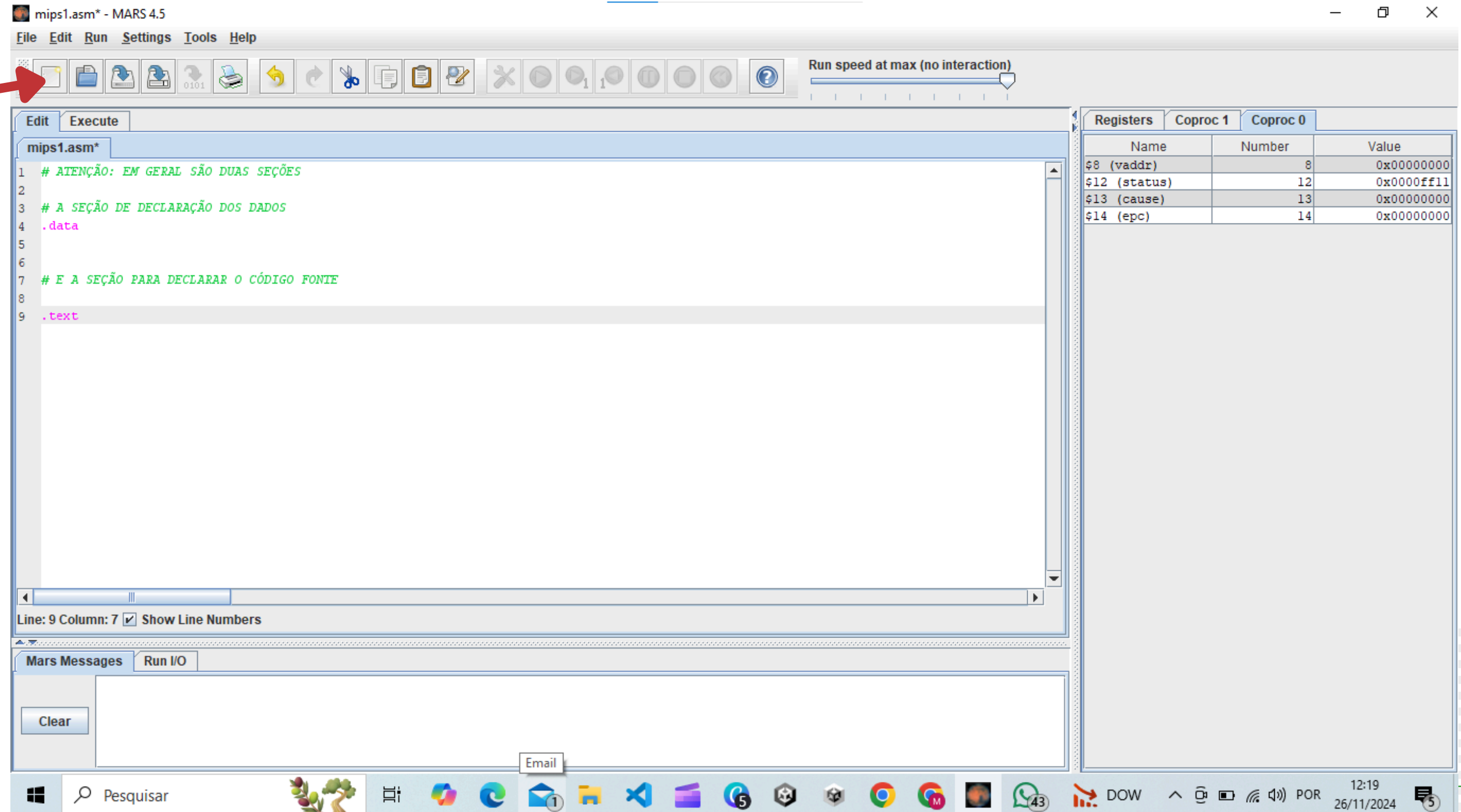
# Função do Coprocessador 0:

- Exceções: Trata erros como divisões por zero ou acessos inválidos à memória.
- Controle de interrupções: Permite que o processador altere seu comportamento dependendo das interrupções externas ou internas.
- Modo de operação: Define se o processador está em modo de usuário (executando um programa normal) ou em modo kernel (acessando recursos do sistema ou tratando exceções).

Portanto, a coluna Coproc0 no MARS 4.5 é útil para observar e depurar eventos de exceção e o controle do sistema, como interrupções e erros, além de fornecer detalhes sobre o status do processador.

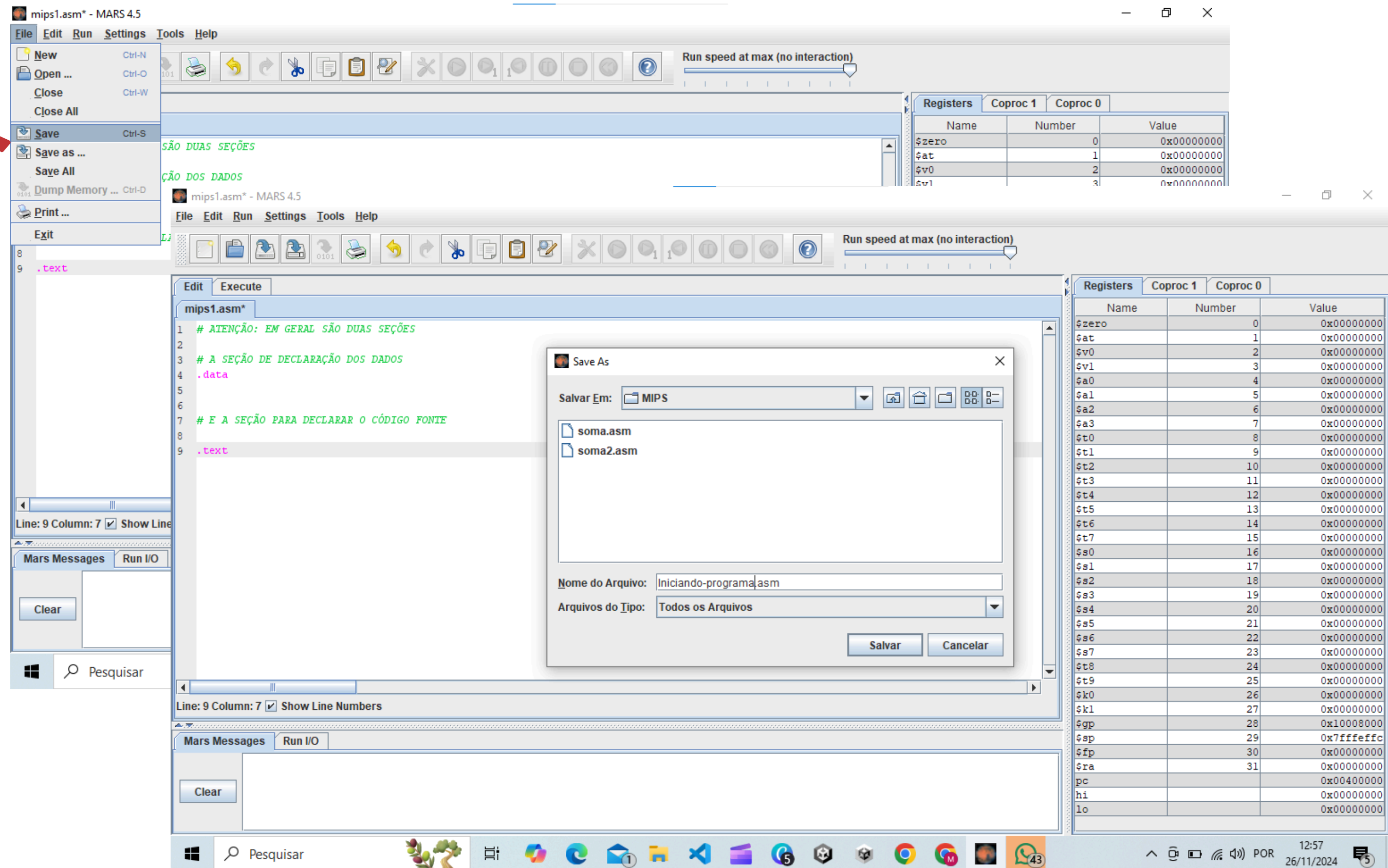
# Iniciando a programação

Clique na opção para criar novo arquivo.



# Salvando o programa

Clique na opção File/Save e escolha o local e clique no botão salvar.



# Exemplo - Soma

```
.data
    prompt1: .asciiz "Digite o primeiro número: "
    prompt2: .asciiz "Digite o segundo número: "
    result_msg: .asciiz "O resultado da soma é: "

.text
    .globl main
main:
    # Solicitar o primeiro número
    li $v0, 4          # código de serviço para print de string
    la $a0, prompt1    # carregar o endereço da mensagem prompt1
    syscall

    li $v0, 5          # código de serviço para ler inteiro
    syscall            # fazer a leitura do número
    move $t0, $v0       # armazenar o primeiro número em $t0

    # Solicitar o segundo número
    li $v0, 4          # código de serviço para print de string
    la $a0, prompt2    # carregar o endereço da mensagem prompt2
    syscall

    li $v0, 5          # código de serviço para ler inteiro
    syscall            # fazer a leitura do número
    move $t1, $v0       # armazenar o segundo número em $t1

    # Realizar a soma
    add $t2, $t0, $t1   # somar os valores em $t0 e $t1 e armazenar o resultado em $t2
```

```
# Mostrar o resultado
li $v0, 4          # código de serviço para print de string
la $a0, result_msg # carregar o endereço da mensagem result_msg
syscall

li $v0, 1          # código de serviço para imprimir inteiro
move $a0, $t2      # mover o resultado da soma para $a0
syscall

# Finalizar o programa
li $v0, 10         # código de serviço para sair
syscall
```

## Funcionamento:

- 1.O programa solicita ao usuário dois números inteiros, utilizando a syscall para imprimir texto (li \$v0, 4) e ler a entrada (li \$v0, 5).
- 2.Os números digitados são armazenados nos registradores \$t0 e \$t1.
- 3.A soma é realizada com a instrução add, e o resultado é armazenado em \$t2.
- 4.O programa então imprime o resultado utilizando a syscall para imprimir um inteiro (li \$v0, 1).
- 5.O programa termina com a syscall para sair (li \$v0, 10).

# Exercícios:

**1. Qual é a função do registrador \$zero na arquitetura MIPS?**

- a) Armazenar o endereço de retorno de uma função.
- b) Armazenar o valor 0 de forma constante.
- c) Armazenar o resultado de operações aritméticas.
- d) Armazenar um valor booleano.



# Exercícios:

**1. Qual é a função do registrador \$zero na arquitetura MIPS?**

a) Armazenar o endereço de retorno de uma função.

**b) Armazenar o valor 0 de forma constante.**

c) Armazenar o resultado de operações aritméticas.

d) Armazenar um valor booleano.

# Exercícios:

**2. Na arquitetura MIPS, qual é o propósito do registrador \$ra?**

- a) Armazenar o endereço de retorno de uma função.
- b) Armazenar o valor de um argumento de função.
- c) Armazenar o resultado de uma operação aritmética.
- d) Armazenar o valor do contador de programa.

# Exercícios:

**2. Na arquitetura MIPS, qual é o propósito do registrador \$ra?**

**a) Armazenar o endereço de retorno de uma função.**

b) Armazenar o valor de um argumento de função.

c) Armazenar o resultado de uma operação aritmética.

d) Armazenar o valor do contador de programa.

# Exercícios:

**3. Quais são os registradores utilizados para operações de ponto flutuante na arquitetura MIPS?**

- a) \$t0 a \$t9.
- b) \$f0 a \$f31.
- c) \$a0 a \$a3.
- d) \$v0 a \$v1.

# Exercícios:

**3. Quais são os registradores utilizados para operações de ponto flutuante na arquitetura MIPS?**

a) \$t0 a \$t9.

**b) \$f0 a \$f31.**

c) \$a0 a \$a3.

d) \$v0 a \$v1.



# Exercícios:

**4. Qual das instruções MIPS é usada para realizar uma operação de soma entre dois registradores de propósito geral?**

- a) add
- b) sub
- c) mul
- d) div
- e) som

# Exercícios:

**4. Qual das instruções MIPS é usada para realizar uma operação de soma entre dois registradores de propósito geral?**

**a) add**

b) sub

c) mul

d) div

e) som

# Exercícios:

## 5. Dada a operação aritmética em MIPS:

`addi $t0, $zero, 5`    #  $\$t0 = 5$

`addi $t1, $zero, 10`    #  $\$t1 = 10$

`add $t2, $t0, $t1`    #  $\$t2 = \$t0 + \$t1$

Qual será o valor armazenado no registrador  $\$t2$  após a execução dessas instruções?

a) 5

b) 10

c) 15

d) 0

e) -5

# Exercícios:

## 5. Dada a operação aritmética em MIPS:

`addi $t0, $zero, 5`    #  $\$t0 = 5$

`addi $t1, $zero, 10`    #  $\$t1 = 10$

`add $t2, $t0, $t1`    #  $\$t2 = \$t0 + \$t1$

Qual será o valor armazenado no registrador  $\$t2$  após a execução dessas instruções?

a) 5

b) 10

**c) 15**

d) 0

e) -5