



UNIVERSIDADE  
FEDERAL DO  
MARANHÃO

# Alocação de Memória

ARTHUR SALIM DA COSTA  
LAIS SILVA COSTA  
MARIA HELENA DE SOUSA COSTA

# Refencias

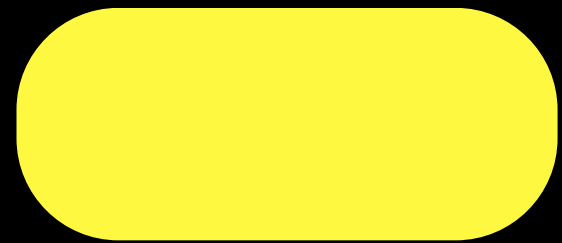
**O código usado tem como base o código mini-so-3.py do prof. Thales Valente**

**Pode ser observado dentro da class SimpleOSSimulated\_v3 nas funções:**

- **def \_\_init\_\_**
- **def allocate\_memory**
- **def free\_memory**
- **def execute\_program**
- **def program\_calc**
- **def program\_echo**

**Pode ser observado também no teste de memória abaixo da classe**

# Legenda



Ação corrente



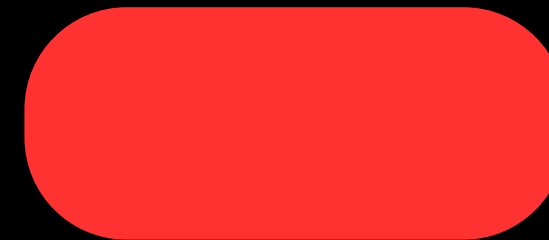
SIM



Condição atendida



NÃO



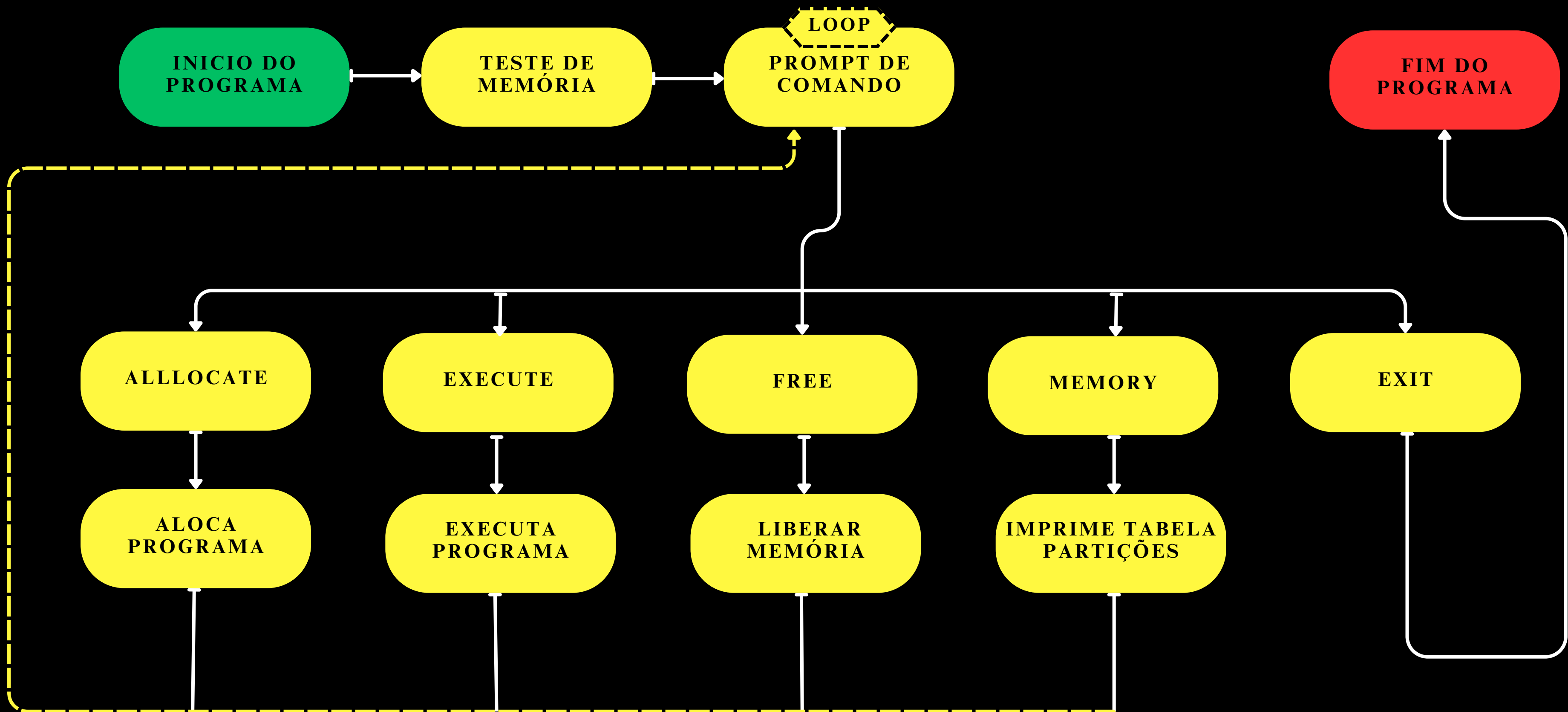
Condição não atendida



Comando em loop

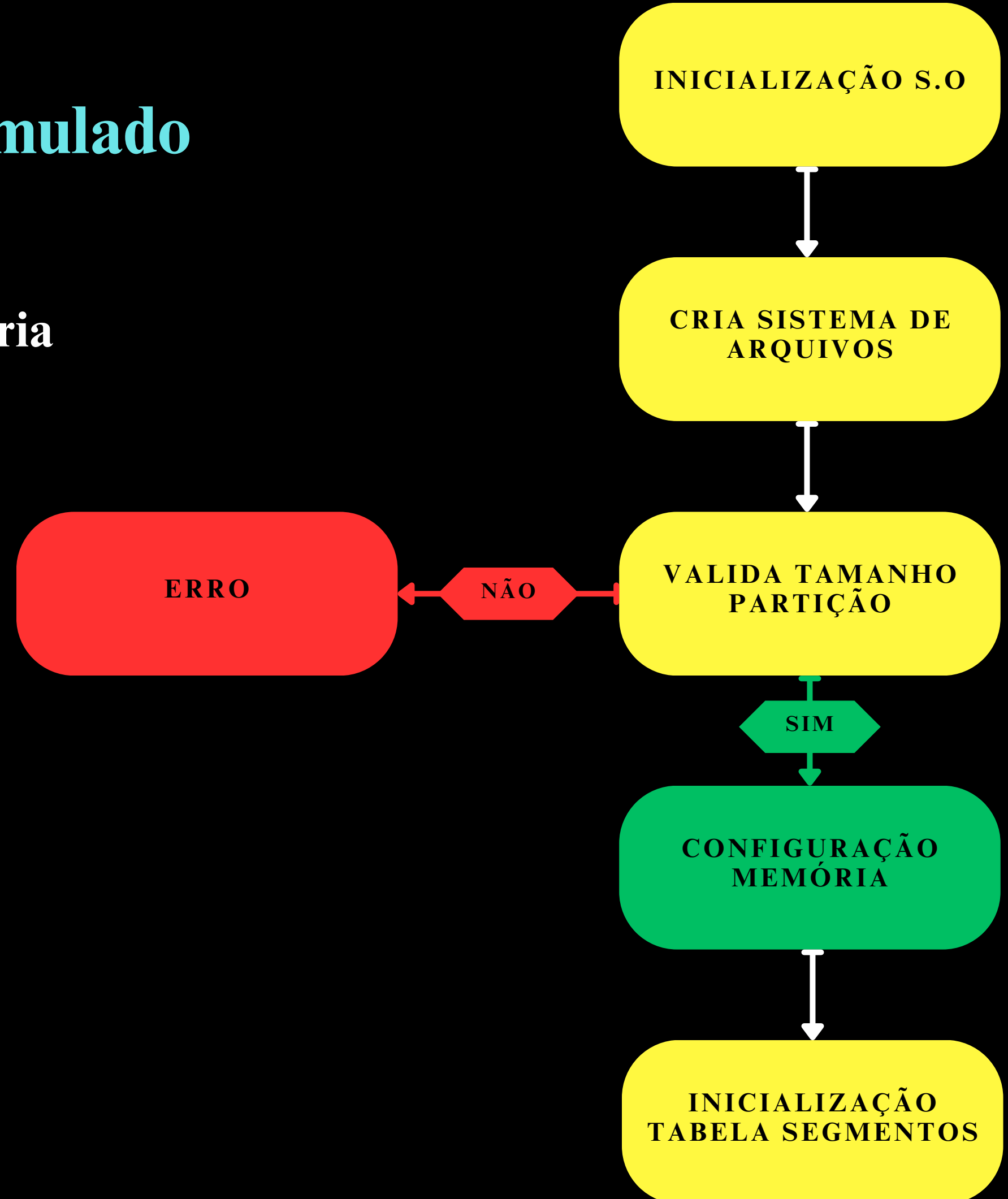


# Visão Geral do programa



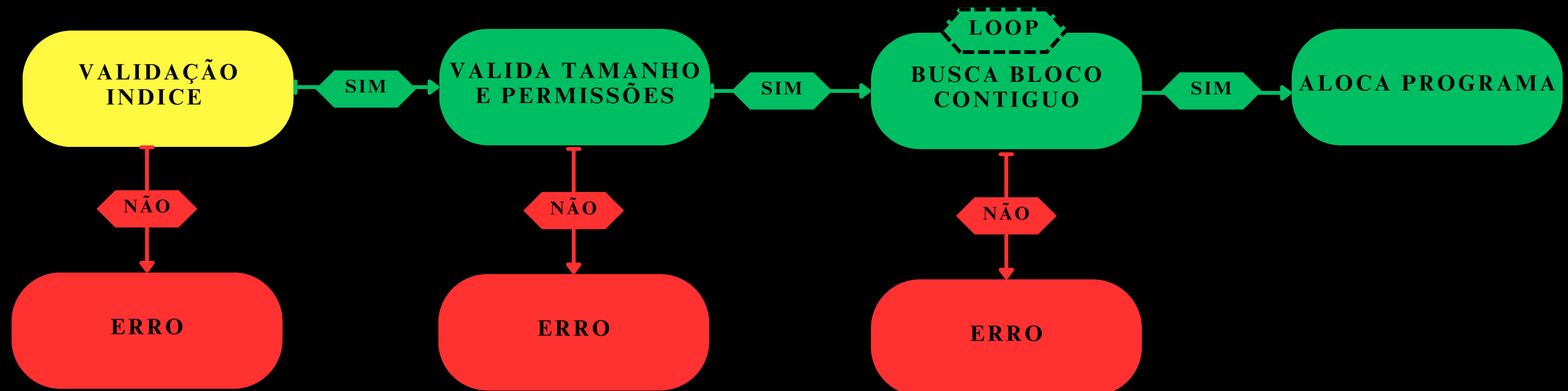
# Início do Sistema Operacional Simulado

1. Configuração da Memória
2. Tabela de Partições e Alocação de Memória
3. Inicialização da Tabela de Segmentos



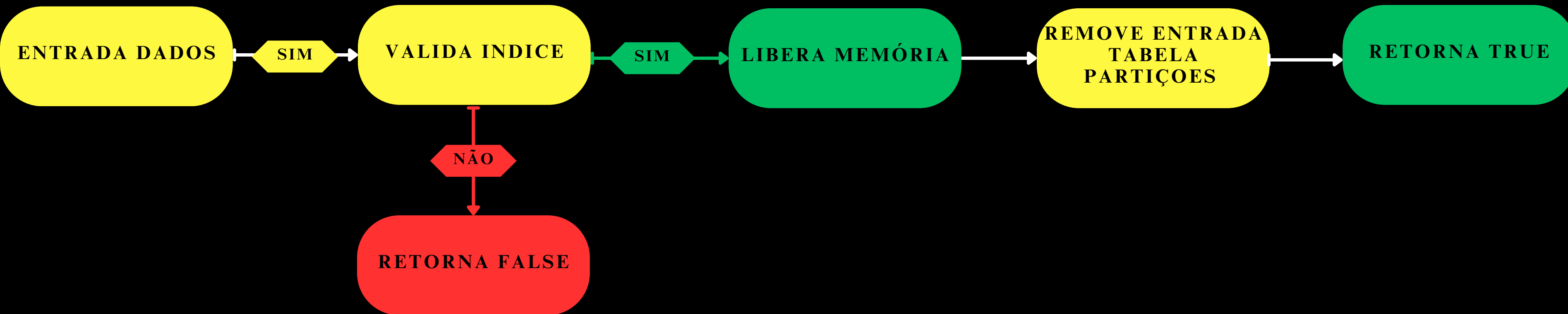
# Alocação Memória

1. Alocação de Memória:
2. Validação de Parâmetros
3. Busca por Bloco Contíguo
4. Processo de Alocação



# Liberação de Memória:

1. Validação dos dados
2. Remoção da Entrada na Tabela de Partições
3. Resultado



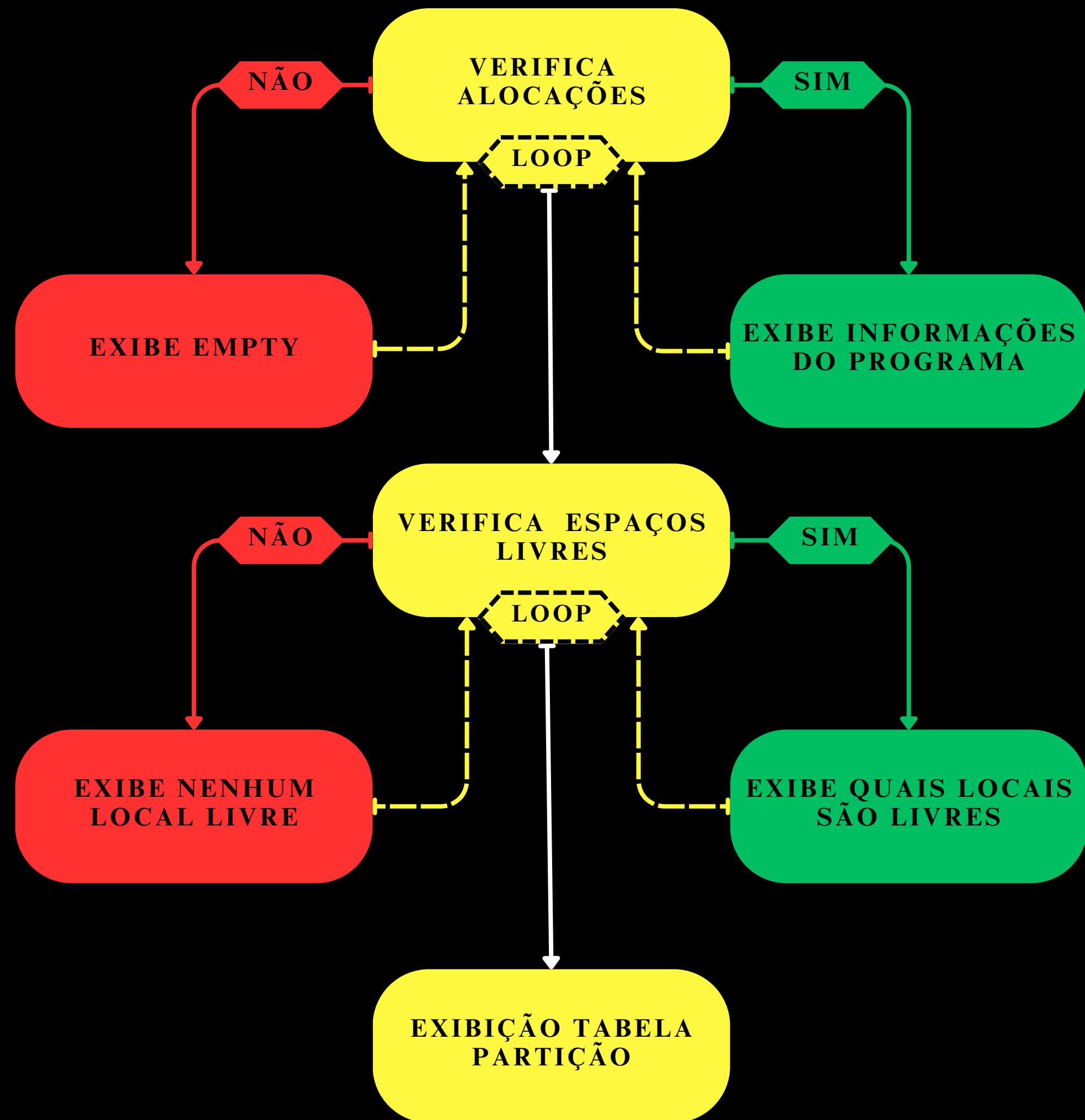
# Impressão da Tabela de Partições

1. Exibição de Programas Alocados

2. Verificação de Programa ou Função

3. Detecção de Locais Livres

4. Exibição da Tabela de Partições





# Impressão da Tabela de Partições

Partições = [4,3,3]

programa echo alocado partição 0

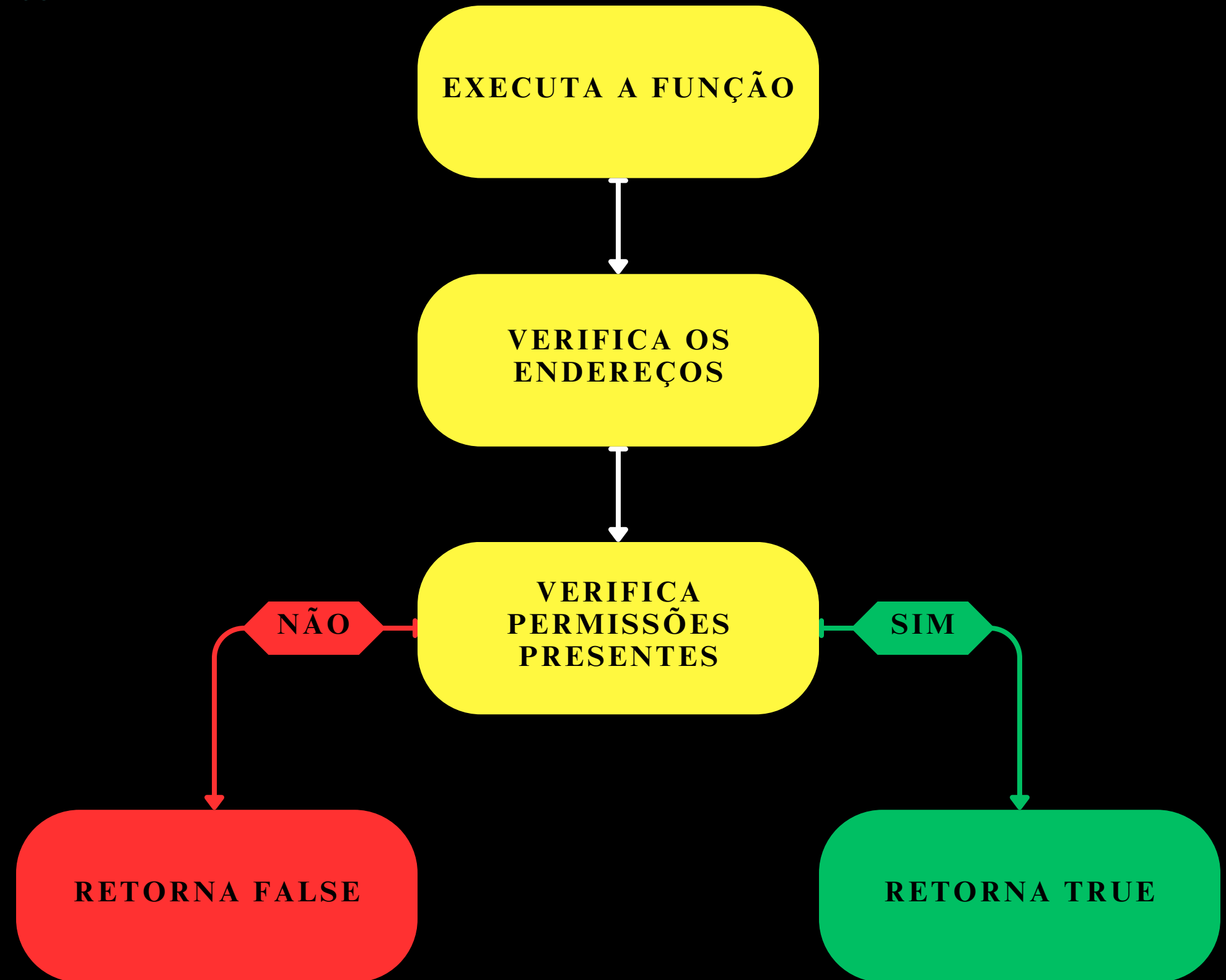
programa calc alocado partição 2

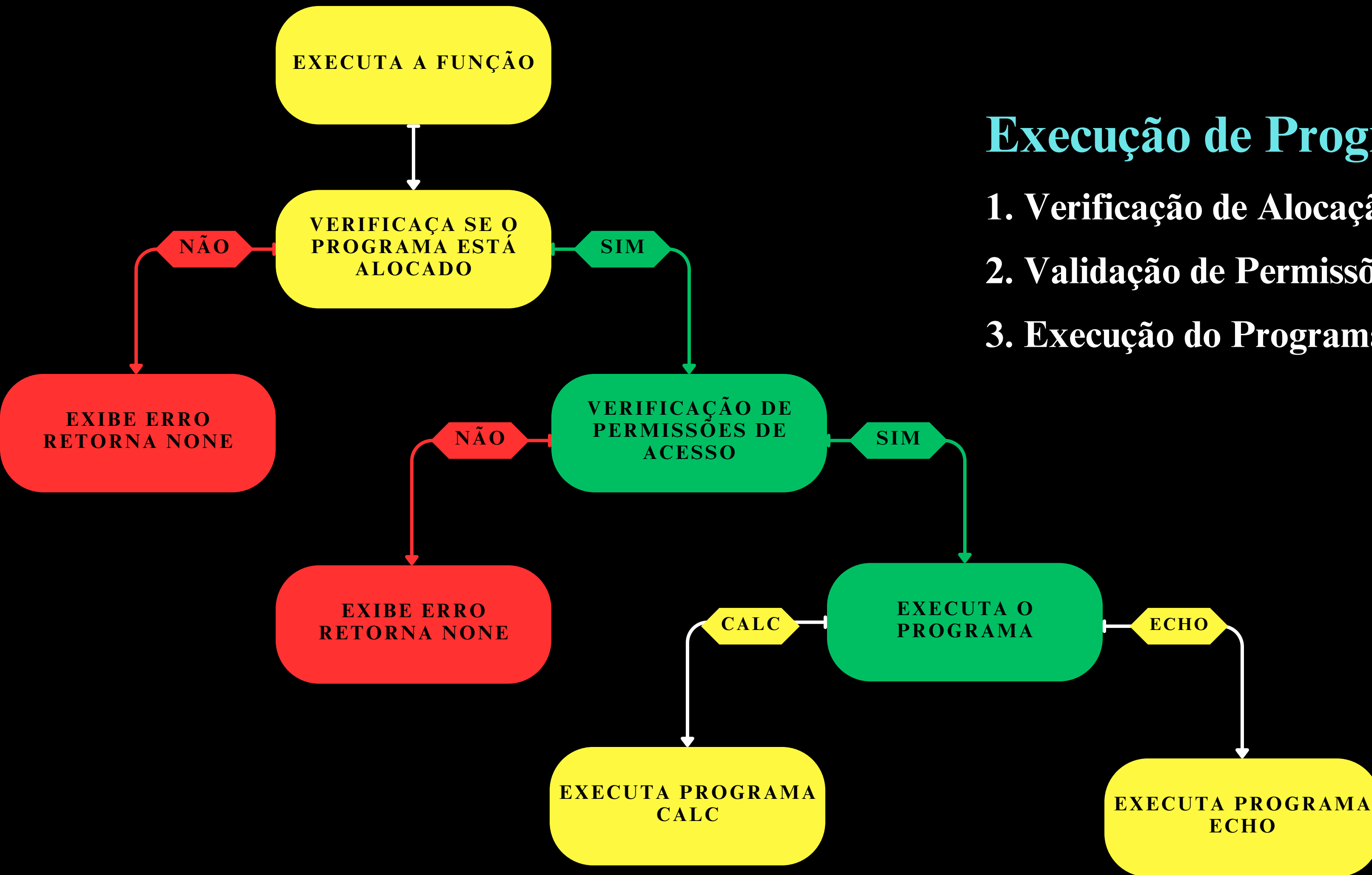
```
=====
Tabela de Partições:
Partição 0: Programa echo, Size: 2
Partição 1: Empty
Partição 2: Programa calc, Size: 3

Locais Livres:
Partição 0: Locais livres nos índices [2, 3]
Partição 1: Locais livres nos índices [0, 1, 2]
Partição 2: Nenhum local livre
=====
```

# Checagem de Acesso à Memória

1. Validação do Índice de Partição
2. Verificação de Intervalo de Endereços
3. Validação de Permissões Requeridas
4. Resultado da Checagem de Acesso



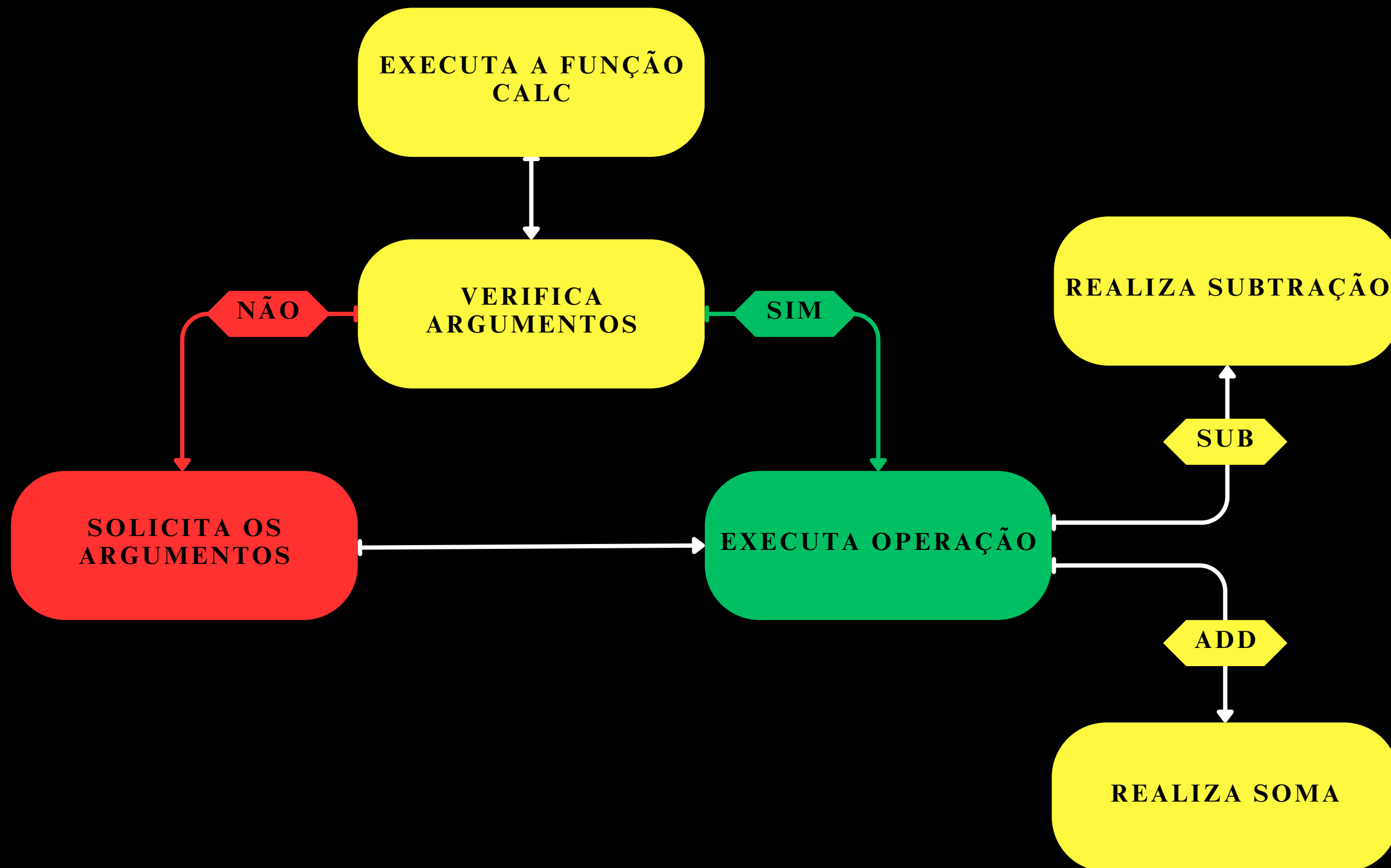


## Execução de Programas

1. Verificação de Alocação na Partição
2. Validação de Permissões de Acesso
3. Execução do Programa

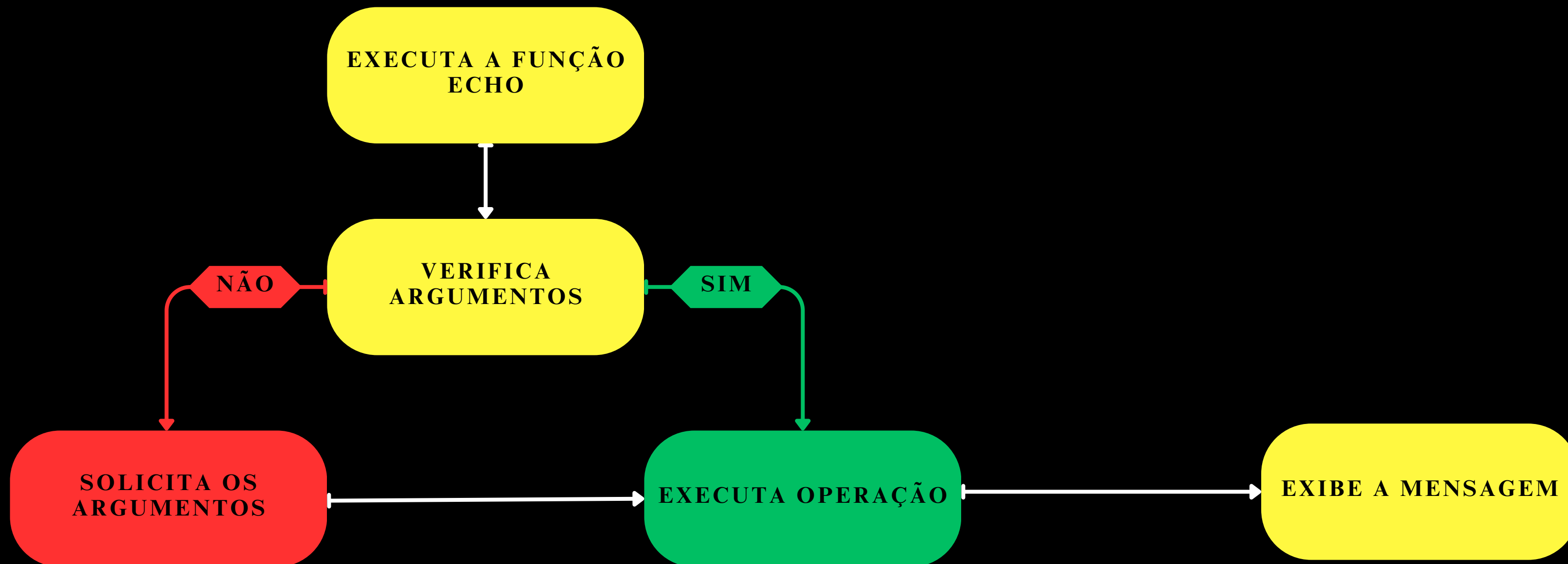
# Programas

## 1. Funcionamento do programa CALC



# Programas

## 1. Funcionamento do programa ECHO



# Programas

## 1. Funcionamento do programa CALC

```
Digite a operação a ser realizada (add, sub): add  
Digite os operandos separados por espaço: 435 234 52  
Resultado da execução: 721
```

```
=====
```

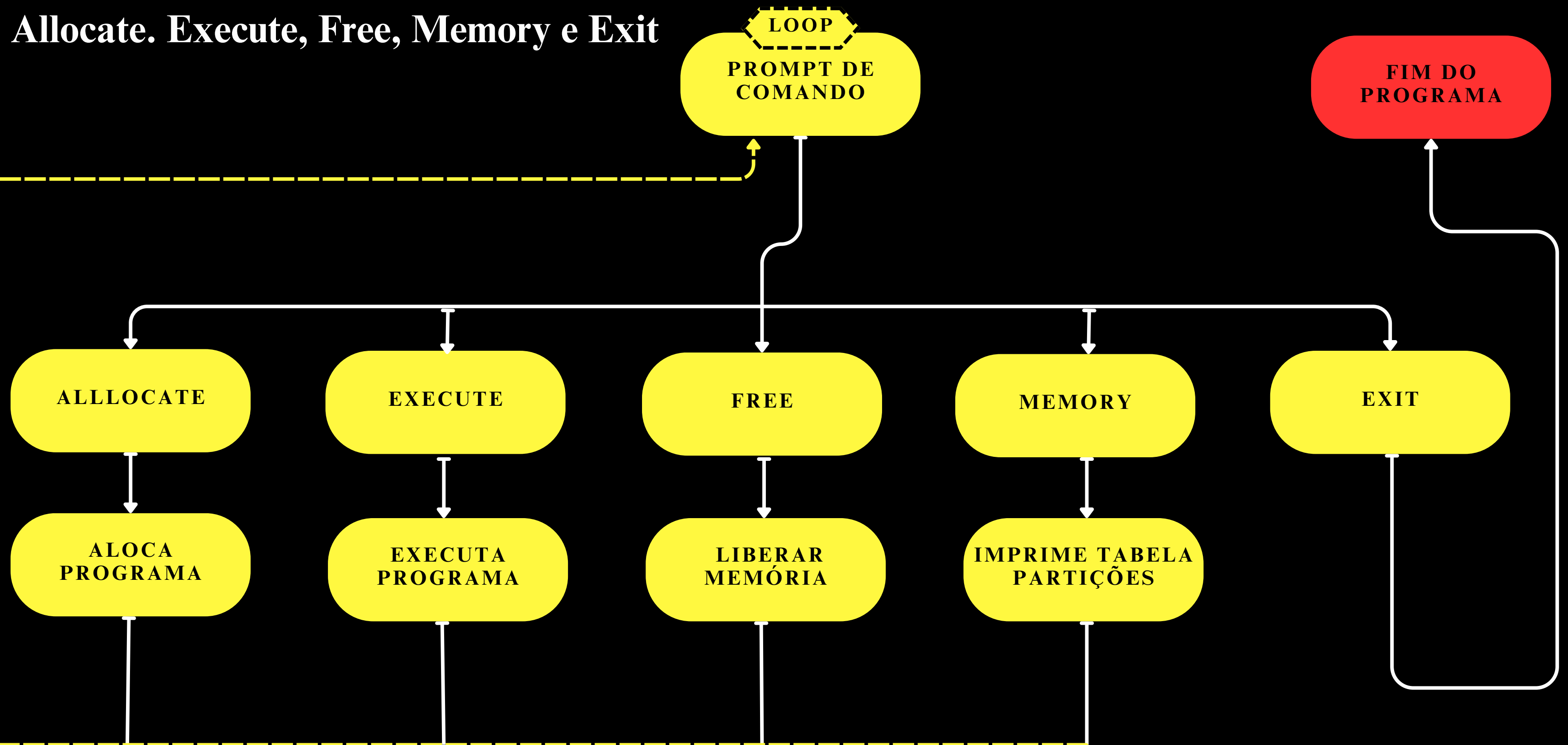
## 1. Funcionamento do programa ECHO

```
Digite o nome do programa a ser executado (calc, echo): echo  
Digite a mensagem a ser ecoada: Hello World! :)  
Resultado da execução: Hello World! :)
```

# Prompt de Comando

Exibição de menu do promp:

Allocate. Execute, Free, Memory e Exit



# Prompt de Comando

## Exibição de menu do promp

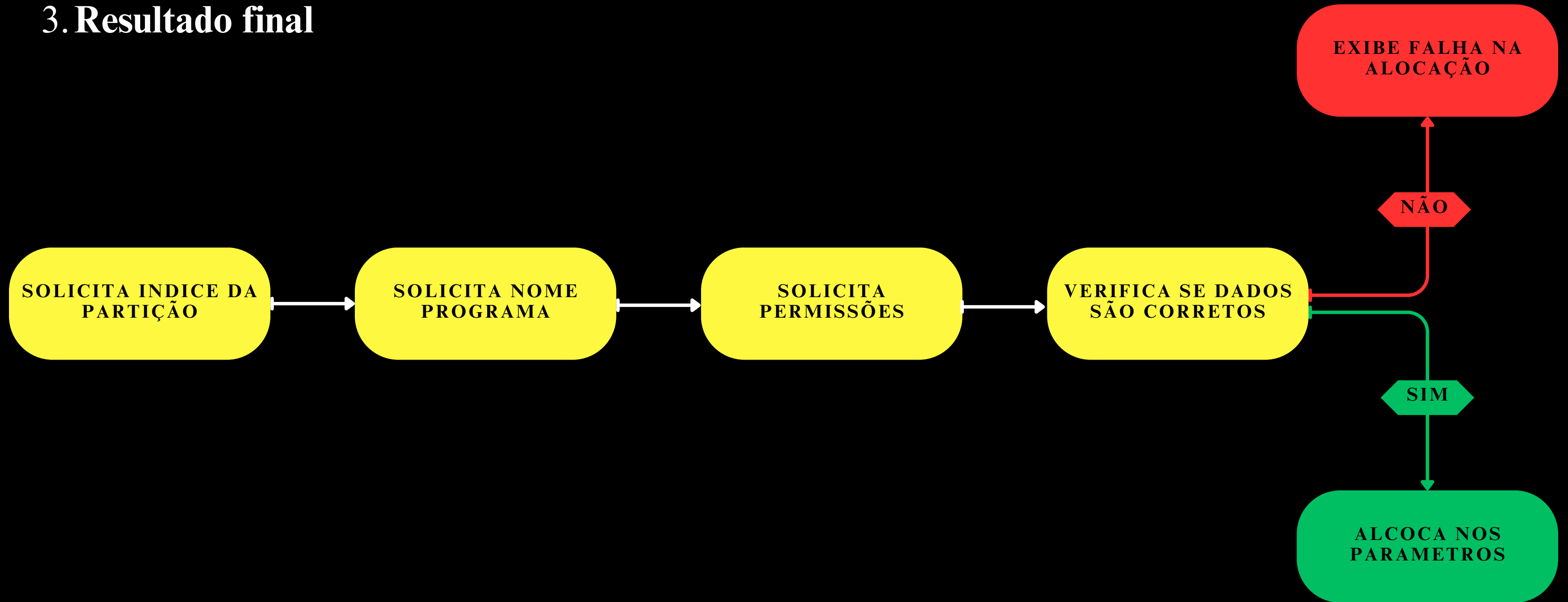
- **ALLOCATE**
- **FREE**
- **EXECUTE**
- **MEMORY**
- **EXIT**

```
=====
                        PROMPT DE COMANDO
=====
Digite um comando (allocate, free, execute, memory, exit): allocate
=====
```



# Comando Allocate

1. Chamada da Função `allocate_memory`
2. Solicita ao usuário dados sobre alocação
3. Resultado final

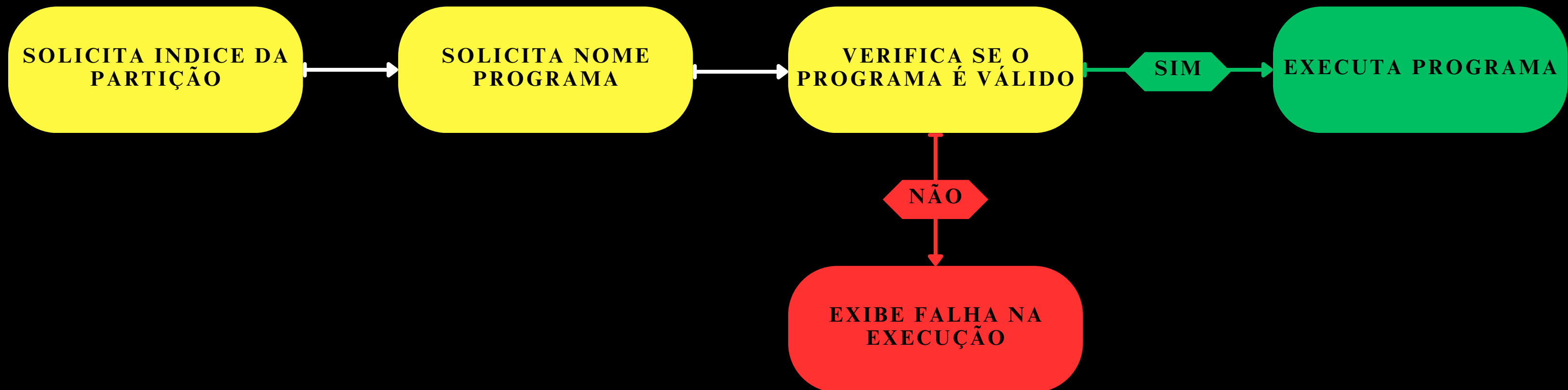


# Comando Allocate

```
=====
                                PROMPT DE COMANDO
=====
Digite um comando (allocate, free, execute, memory, exit): allocate
=====
Digite o índice da partição (0, 1, 2): 0
Digite o nome do programa: calc
Digite o tamanho do programa: 3
Digite as permissões (r, w, x): rwX
Memória alocada com sucesso: Partição 0, Início 0, Tamanho 3, Permissões: rwX
                                PROMPT DE COMANDO
=====
Digite um comando (allocate, free, execute, memory, exit): allocate
=====
Digite o índice da partição (0, 1, 2): 2
Digite o nome do programa: echo
Digite o tamanho do programa: 3
Digite as permissões (r, w, x): rx
Falha ao alocar memória.
```

# Comando Execute

1. Solicita ao usuário a partição desejada
2. Solicita o nome do programa a ser executado
3. Resultado da execução



# Comando Execute

## PROMPT DE COMANDO

=====

Digite um comando (allocate, free, execute, memory, exit): *execute*

=====

Digite o índice da partição (0, 1, 2): *0*

Digite o nome do programa a ser executado (calc, echo): *calc*

Digite a operação a ser realizada (add, sub): *sub*

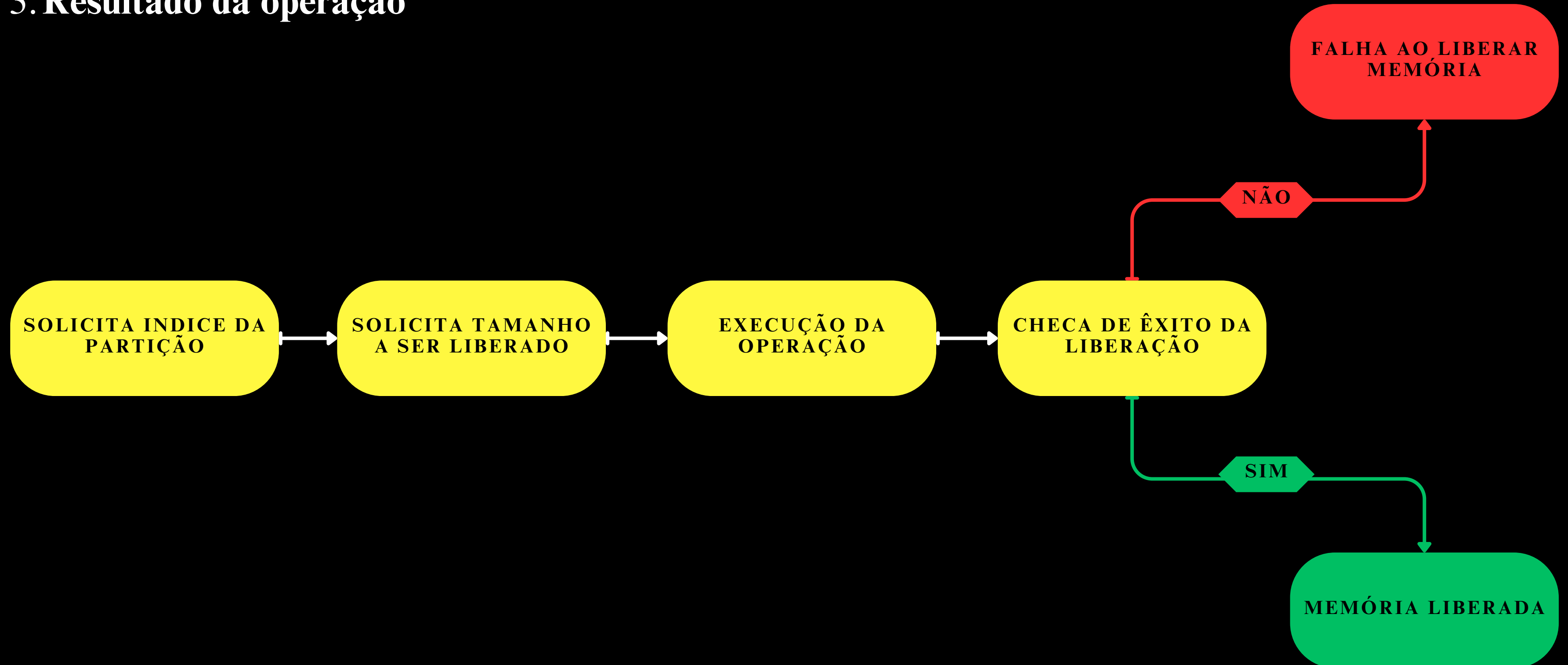
Digite os operandos separados por espaço: *999 323 123*

Resultado da execução: 553

=====

# Comando Free

1. Solicita ao usuário dados para liberação de memória
2. Verifica os dados fornecidos
3. Resultado da operação



# Comando Free

1. Solicita ao usuário dados para liberação de memória
2. Verifica os dados fornecidos
3. Resultado da operação

```
=====
Digite um comando (allocate, free, execute, memory, exit): free
=====
Digite o índice da partição (0, 1, 2): 0
Digite o tamanho do bloco a ser liberado: 3
Memória liberada com sucesso: Partição 0, Tamanho 3.

                        PROMPT DE COMANDO

=====
Digite um comando (allocate, free, execute, memory, exit): free
=====
Digite o índice da partição (0, 1, 2): 2
Digite o tamanho do bloco a ser liberado: 2
Memória liberada com sucesso: Partição 2, Tamanho 2.
```

# Comando Memory

1. chama

2. Exibe Tabela de partições

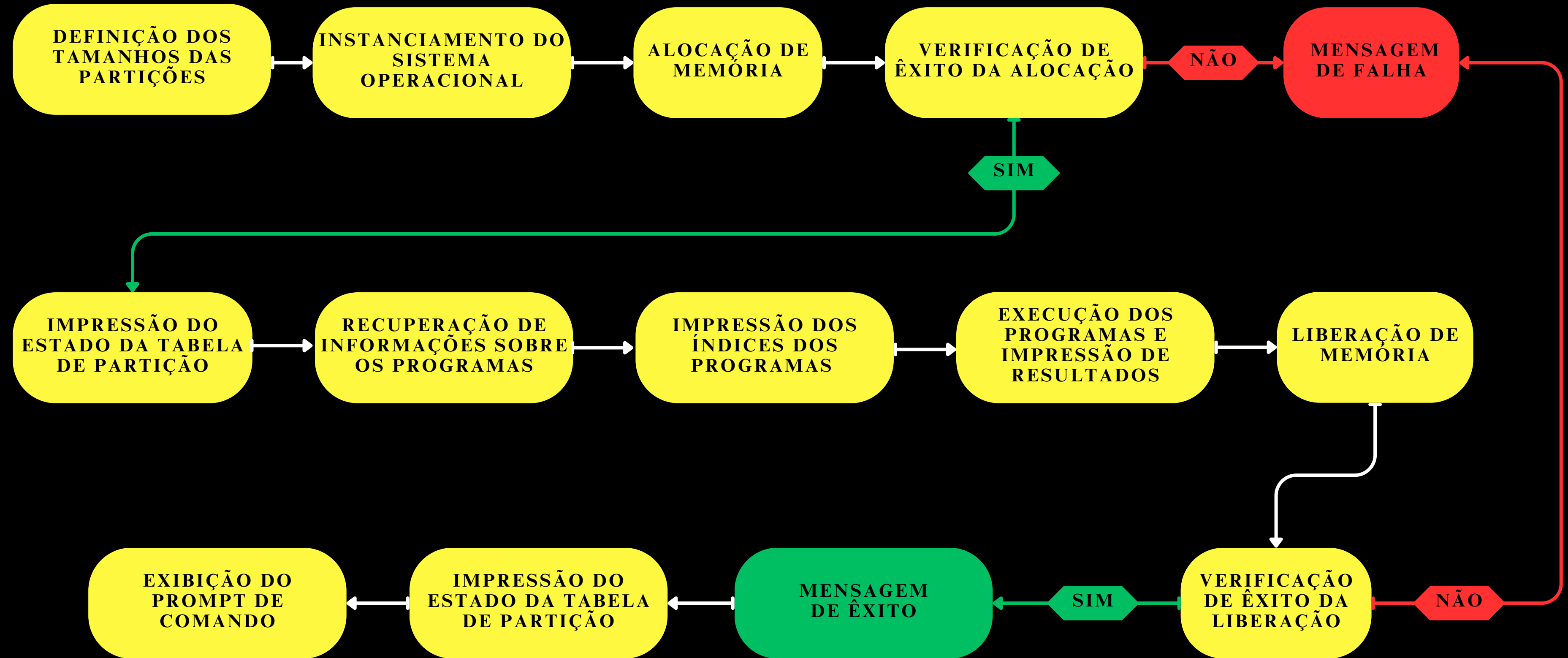
# Comando Exit

1. Encerra o programa

```
=====
                                PROMPT DE COMANDO
=====
Digite um comando (allocate, free, execute, memory, exit): memory
=====
Tabela de Partições:
Partição 0: Programa calc, Size: 3
Partição 1: Programa echo, Size: 3
Partição 2: Empty

Locais Livres:
Partição 0: Locais livres nos índices [3]
Partição 1: Nenhum local livre
Partição 2: Locais livres nos índices [0, 1, 2]
=====
                                PROMPT DE COMANDO
=====
Digite um comando (allocate, free, execute, memory, exit): exit
Sistema encerrado.
```

# TESTE DE FUNCIONAMENTO





## **Reconhecimentos e Direitos Autorais**

**@autor: MARIA HELENA DE SOUSA COSTA  
LAIS SILVA COSTA  
ARTHUR SALIM DA COSTA**

**@contato: maria.hsc@discente.ufma.br  
lais.sc@discente.ufma.br  
arthur.salim@discente.ufma.br**

**@data última versão: [10.12.2023]**

**@versão: 1.0**

**@Agradecimentos: Universidade Federal do Maranhão (UFMA), Professor Doutor Thales Levi Azevedo Valente, e colegas de curso.**

**@Copyright/License**

**Este material é resultado de um trabalho acadêmico para a disciplina SISTEMAS OPERACIONAIS, sobre a orientação do professor Dr. THALES LEVI AZEVEDO VALENTE, semestre letivo 2023.2, curso Engenharia da Computação, na Universidade Federal do Maranhão (UFMA). Todo o material sob esta licença é software livre: pode ser usado para fins acadêmicos e comerciais sem nenhum custo. Não há papelada, nem royalties, nem restrições de "copyleft" do tipo GNU. Ele é licenciado sob os termos da licença MIT reproduzida abaixo e, portanto, é compatível com GPL e também se qualifica como software de código aberto. É de domínio público. Os detalhes legais estão abaixo. O espírito desta licença é que você é livre para usar este material para qualquer finalidade, sem nenhum custo. O único requisito é que, se você usá-los, nos dê crédito.**

**Copyright © 2023 Educational Material**

**Este material está licenciado sob a Licença MIT. É permitido o uso, cópia, modificação, e distribuição deste material para qualquer fim, desde que acompanhado deste aviso de direitos autorais.**

**O MATERIAL É FORNECIDO "COMO ESTÁ", SEM GARANTIA DE QUALQUER TIPO, EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS NÃO SE LIMITANDO ÀS GARANTIAS DE COMERCIALIZAÇÃO, ADEQUAÇÃO A UM DETERMINADO FIM E NÃO VIOLAÇÃO. EM HIPÓTESE ALGUMA OS AUTORES OU DETENTORES DE DIREITOS AUTORAIS SERÃO RESPONSÁVEIS POR QUALQUER RECLAMAÇÃO, DANOS OU OUTRA RESPONSABILIDADE, SEJA EM UMA AÇÃO DE CONTRATO, ATO ILÍCITO OU DE OUTRA FORMA, DECORRENTE DE, OU EM CONEXÃO COM O MATERIAL OU O USO OU OUTRAS NEGOCIAÇÕES NO MATERIAL.**

**Para mais informações sobre a Licença MIT: <https://opensource.org/licenses/MIT>.**