Abstrações de Memória:
Memória Física e Espaços de Endereçamento

Prof. Dr. Márcio Castro
marcio.castro@ufsc.br

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

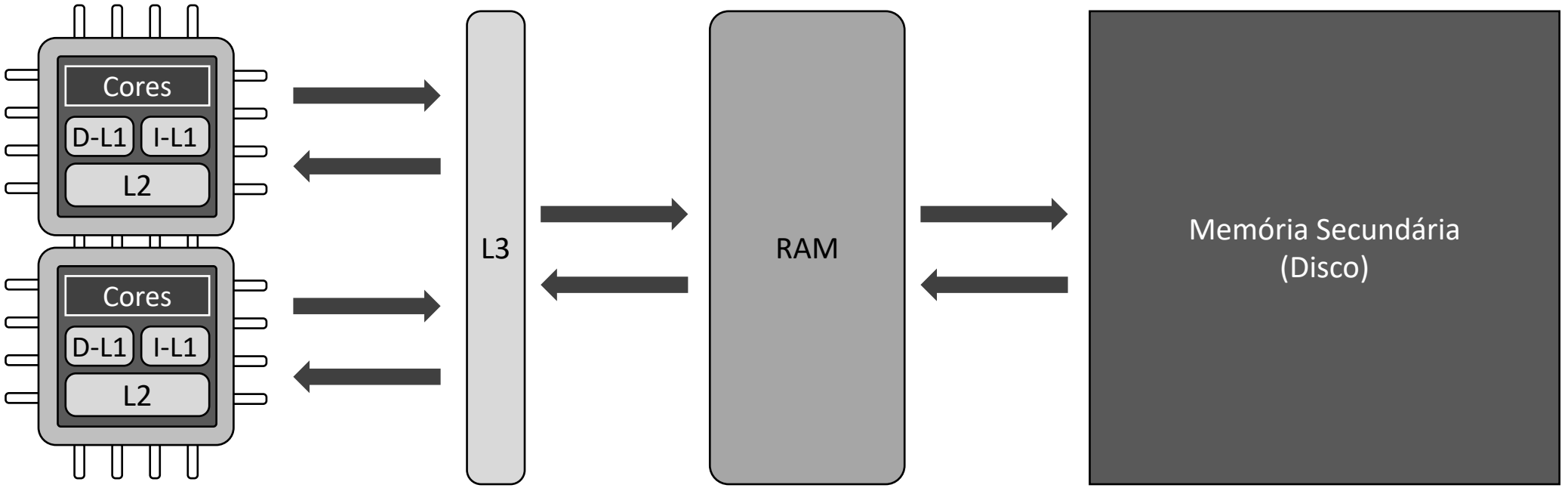
1 Introdução

Introdução

- A memória é um recurso importante que deve ser gerenciado pelo SO
- **Programadores gostariam que a memória do computador fosse:**
  - ~~Privada~~ → A memória física (RAM) é compartilhada
  - ~~De tamanho infinito~~ → Possui tamanho limitado
  - ~~Extremamente rápida~~ → Muito mais lenta do que o processador
  - ~~Fosse não-volátil~~ → Perda dos dados quando o computador é desligado
  - ~~Extremamente barata~~ → Memórias de alto desempenho são caras

Introdução

- **Solução atual:** hierarquia de memória
  - Alguns **MBs** de memórias voláteis extremamente **rápidas** e **caras** (**caches**)
  - Alguns **GBs** de memórias voláteis de **velocidade e custo médio** (**RAM**)
  - Alguns **TBs** de memórias não voláteis **lentas** e **baratas** (**discos**)



Introdução

- A função do SO é **abstrair** essa hierarquia em um **modelo útil** e, então, **gerenciar essa abstração**

- **Gerenciador de memória**
  - Parte do SO responsável por **gerenciar a memória** do computador
  - Mantém informações sobre quais partes da memória estão em **uso** ou **livres**
  - **Aloca** e **desaloca** memória dos processos

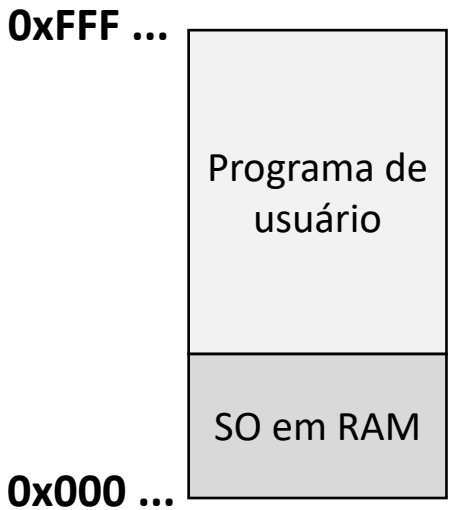
Introdução

- **Níveis de abstração de memória**
  - Sem abstração
  - Espaços de endereçamento
  - Memória virtual

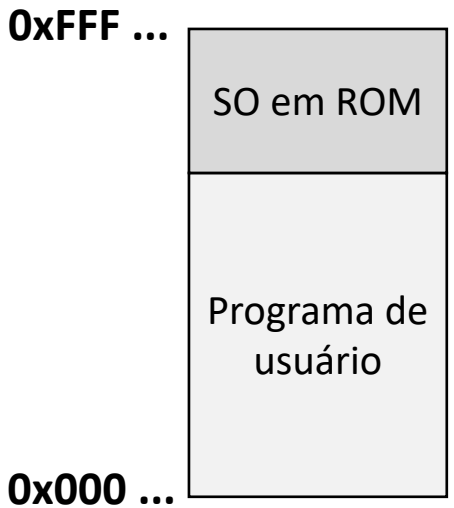
2 Gestão de memória sem abstração

Gestão de memória sem abstração

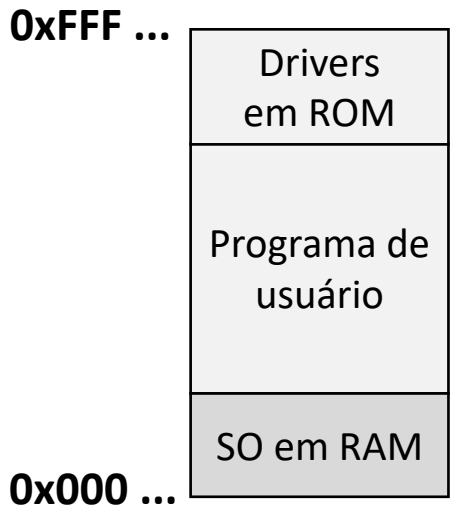
- A **memória física** é diretamente **exposta** ao programador
- Conjunto de endereços de memória de **0** até o **limite de tamanho da memória**
- Utilizado em SOs **sem multiprogramação** (antes da década de 80)



Mainframes



Sistemas embarcados

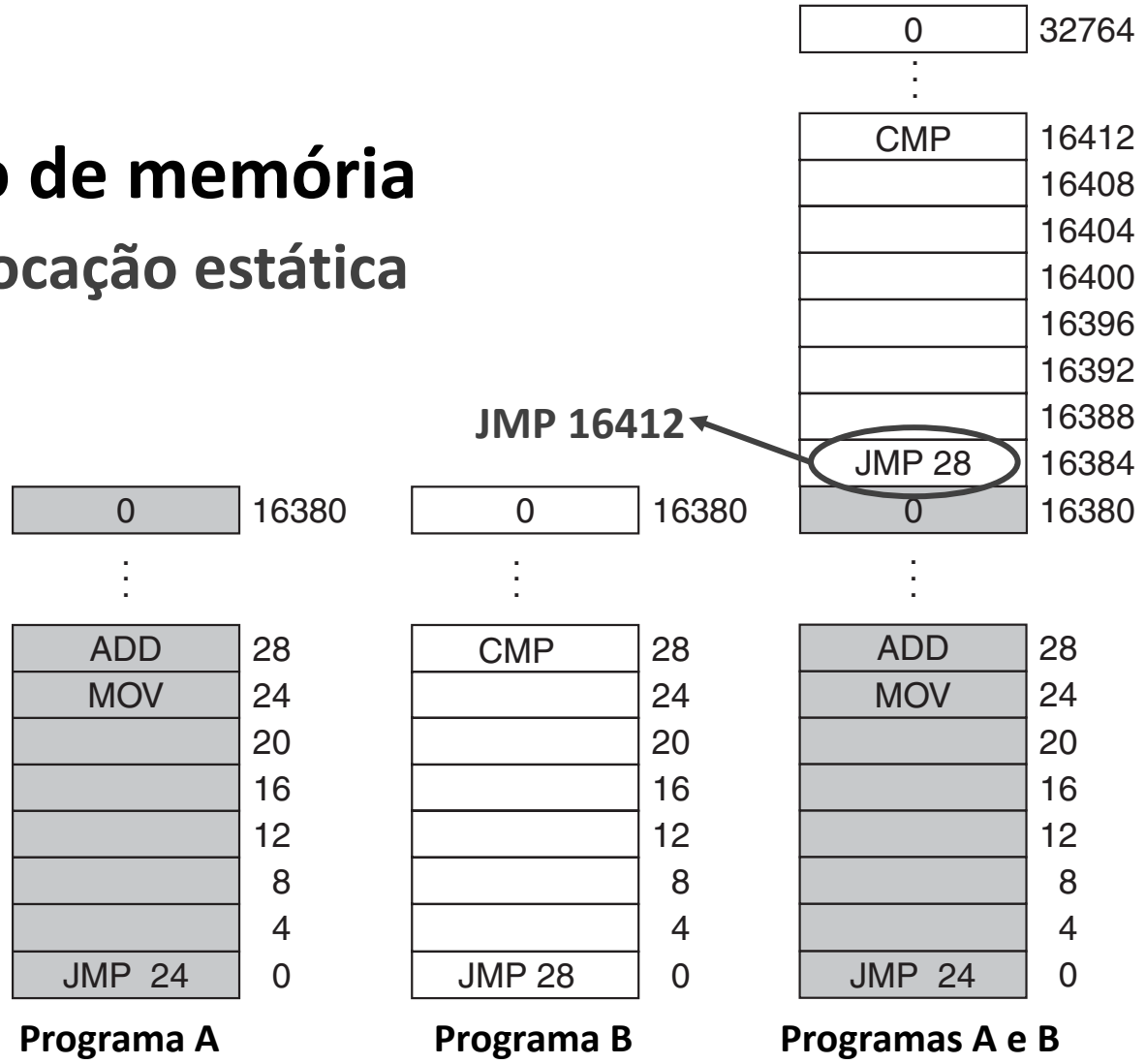


Primeiros PCs

Gestão de memória sem abstração

- **Múltiplos programas sem abstração de memória**
  - Necessidade de uso da técnica de **realocação estática**

- **Exemplo**
  - Dois programas **A** e **B** de 16 KB
  - Endereços utilizados nas instruções do programa **B** precisam ser realocados
  - **Realocação estática**: soma-se o valor **16.384** em cada endereço de **B**



Gestão de memória sem abstração

- **Problemas da gestão de memória sem abstração**
  - Programas podem acidentalmente (ou intencionalmente) **danificar o SO**, exceto se algum mecanismo específico de proteção for usado
  - É **difícil executar e gerenciar** múltiplos programas simultaneamente na memória

3 Espaços de endereçamento

Espaços de endereçamento

- **Dois problemas precisam ser resolvidos para facilitar a execução de múltiplos programas**
  - **Proteção** de memória
  - **Realocação** de instruções na memória

- **Espaços de endereçamento**
  - Uma **memória abstrata** para os processos
  - **Conjunto de endereços** que o processo pode usar para **endereçar a memória**
  - Cada processo tem sem **próprio espaço de endereçamento**

Espaços de endereçamento

- **O conceito de espaço de endereçamento é bastante geral e ocorre em diversos contextos**
  - Se um número de telefone possui **10 dígitos** (2 dígitos da área + 8 dígitos), seu espaço de endereçamento varia de **0000000000** até **9999999999**
  - Endereços **IPv4** possuem **32 bits**, logo, seu espaço de endereçamento varia de **0** até $2^{32} - 1$

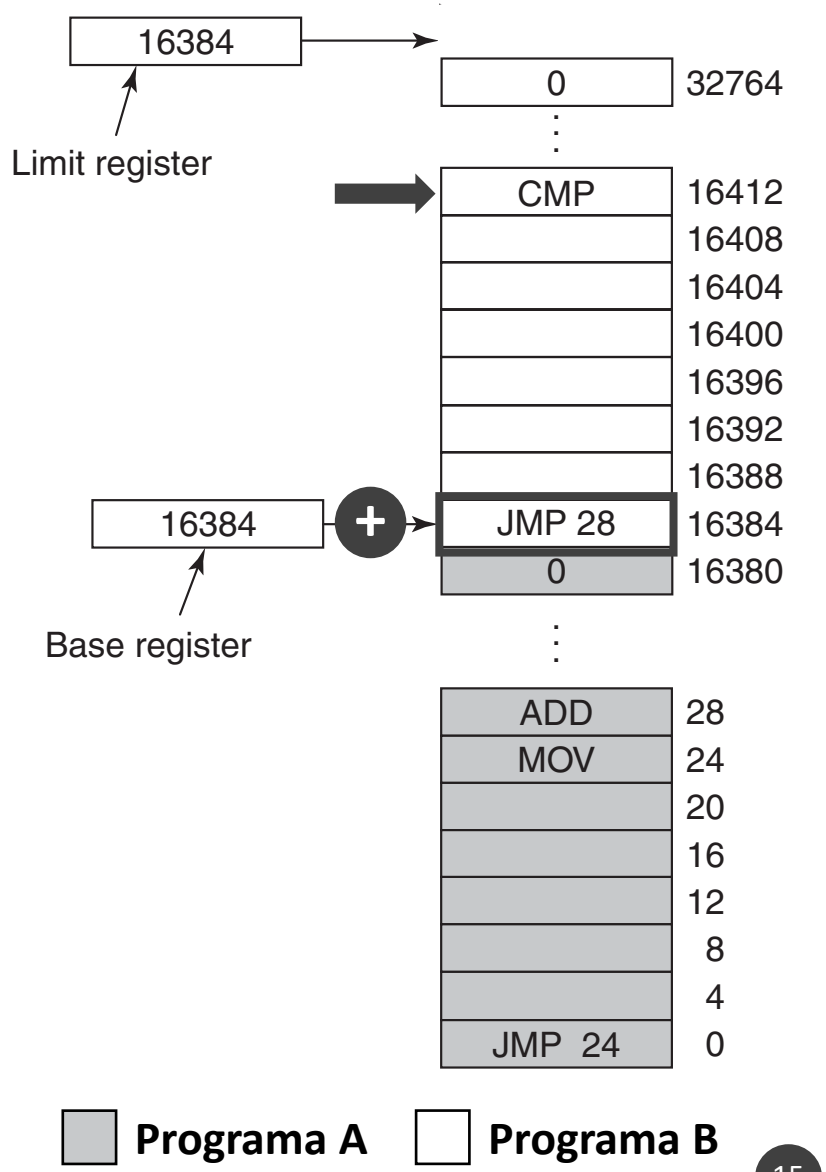
- **No contexto do gerenciamento de memória**
  - Cada processo deverá ter **seu próprio espaço de endereçamento**
  - **Um mesmo endereço X** em dois processo **A** e **B** deverá estar em **localizações diferentes na memória física**

Espaços de endereçamento

- **Solução:** registradores **base** e **limite**
  - Programas são carregados em **porções contíguas** da memória **sem necessidade de realocação estática**
  - **Registrador base:** possui o **primeiro endereço físico** onde o programa reside
  - **Registrador limite:** armazena o **tamanho do programa**
- Os valores dos registrados **base** e **limite** fazem parte do **contexto do processo**
  - Quando um processo é **escalonado**, o seu contexto é **restaurado nos registrados do processador**
  - Os registradores **base** e **limite** serão **atualizados com valores referentes ao processo**

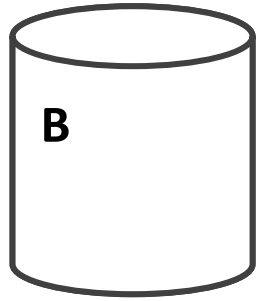
Espaços de endereçamento

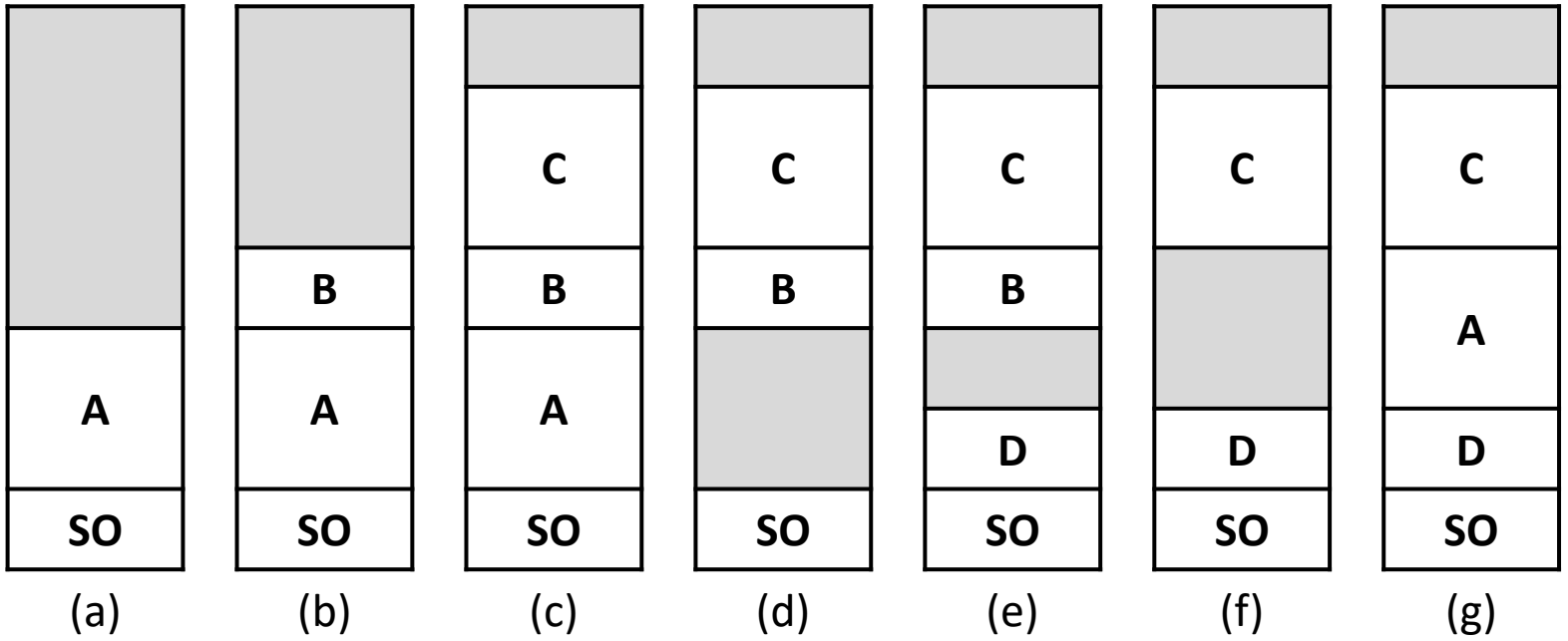
- **Funcionamento:**
  - Instrução faz referência à memória
  - O *hardware* **adiciona o valor base** ao endereço requisitado antes de enviar para o barramento
  - O *hardware* verifica se o endereço está **dentro do limite**
- **Exemplo:**
  - Programa **B** executa instrução **JMP 28**
  - O *hardware* faz a operação **28 + 16.384 = 16.412**
  - Como **16.412 ≤ 32.764**, a informação é enviada para o barramento



Swapping

- Transferência de programas entre disco e RAM
- **Utilização do disco como extensão da RAM**
  - **Swap-in**: Disco → RAM
  - **Swap-out**: RAM → Disco





Processos em memória

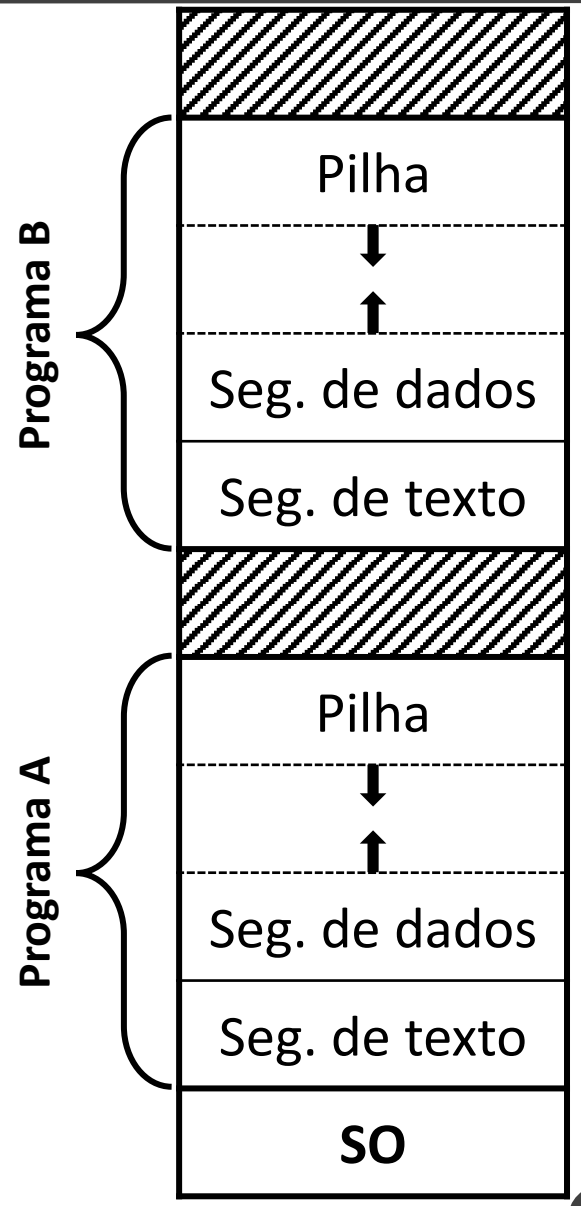
- **Processos com tamanho fixo**
  - Simples de gerenciar
  - Previsibilidade

- **Processos com tamanho variável**
  - Alocação dinâmica de dados
  - Espaço extra para permitir crescimento

Processos em memória

- **Processos com tamanho variável**
  - **Espaço para crescimento** dentro do espaço de endereçamento do processo

- **Partes de um programa em memória**
  - **Segmento de texto**: instruções do programa
  - **Segmento de dados**: alocação estática + dinâmica de dados
  - **Pilha**: variáveis locais e endereços de retorno de funções



Obrigado pela atenção!

Dúvidas? Entre em contato:

- marcio.castro@ufsc.br
- www.marciocastro.com

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

LaPeSD

Distributed Systems Research Lab

www.lapesd.inf.ufsc.br