

# Sistemas Distribuídos - ESP625

Prof<sup>a</sup> Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação  
Instituto Federal de Educação, Ciência e Tecnologia da Bahia  
Campus de Feira de Santana

*carolsoko@ifba.edu.br*

July 25, 2024

# Projeto de Sistemas de Tempo Real

- 1 Projeto de Sistemas de Tempo Real
- 2 Hardware para Sistemas de Tempo Real
- 3 Arquitetura Básica de um Sistema de Tempo Real
  - Arquitetura Básica de um Sistema Computacional
  - Arquitetura com DMA
  - Pipeline
  - CISC X RISC
  - Processador
- 4 Conclusão
- 5 Referências

# Projeto de Sistemas de Tempo Real

# Projeto de Sistemas de Tempo Real

Deve-se estar atento para:

Seleção do Hardware e do Software

- Relação Custo X Benefício

- Como tratar paralelismo, sincronização e distribuição.
- Herda problemas típicos de sistemas distribuídos
- Pode incrementar desempenho e confiabilidade.

# Projeto de Sistemas de Tempo Real

Deve-se estar atento para:

Seleção do Hardware e do Software

- Relação Custo X Benefício

- Como tratar paralelismo, sincronização e distribuição.
- Herda problemas típicos de sistemas distribuídos
- Pode incrementar desempenho e confiabilidade.

Especificação e Projeto de Sistema de Tempo Real:

- Representação correta do comportamento funcional e temporal do sistema. [Burns e Wellings 1997]

# Projeto de Sistemas de Tempo Real

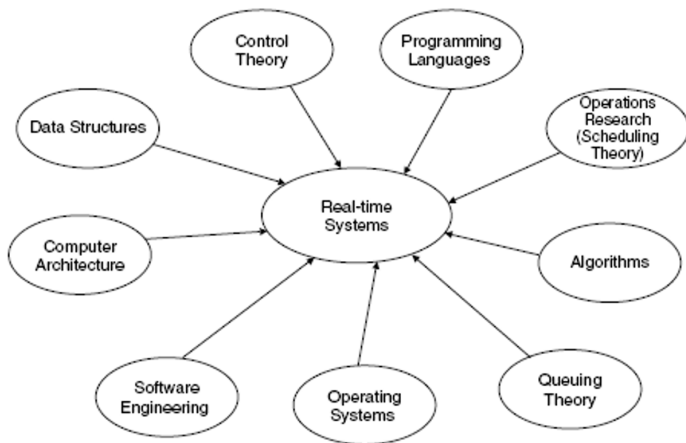
Deve-se estar atento para:

Entendimento das nuances das linguagens de programação e implicações para o Sistema de Tempo Real:

- Abstração dos dispositivos
- Abstrações temporais
- Representação em termos de código de máquina

Cuidados com prevenção e tolerância a falhas

# Projeto de Sistemas de Tempo Real



# Hardware para Sistemas de Tempo Real



# Hardware para Sistemas de Tempo Real

## Aspectos interessantes:

- Os hardwares tradicionais são imprevisíveis, pois possuem recursos como interrupções, memória cache, entre outros, que podem falhar ou demorar demais, simulando falhas.

# Hardware para Sistemas de Tempo Real

## Aspectos interessantes:

- Os hardwares tradicionais são imprevisíveis, pois possuem recursos como interrupções, memória cache, entre outros, que podem falhar ou demorar demais, simulando falhas.
- O problema do projeto de hardware está na determinação do **Worst Case Execution Time (WCET)** também chamado de Tempo de Execução no Pior Caso.

# Hardware para Sistemas de Tempo Real

## Aspectos interessantes:

- Os hardwares tradicionais são imprevisíveis, pois possuem recursos como interrupções, memória cache, entre outros, que podem falhar ou demorar demais, simulando falhas.
- O problema do projeto de hardware está na determinação do **Worst Case Execution Time (WCET)** também chamado de Tempo de Execução no Pior Caso.
- Sem o conhecimento exato do **WCET**, não podemos prever se o sistema executará corretamente, pois a tarefa pode consumir mais tempo do que o declarado, fazendo com que outras tarefas percam seus **Dealines**. [Burns e Wellings 1997]

# Hardware para Sistemas de Tempo Real

## Itens a serem observados:

O hardware deve estar totalmente projetado para operar um sistema de tempo-real.

# Hardware para Sistemas de Tempo Real

## Itens a serem observados:

O hardware deve estar totalmente projetado para operar um sistema de tempo-real.

- Conjunto de instruções (simples ou complexas?).

# Hardware para Sistemas de Tempo Real

## Itens a serem observados:

O hardware deve estar totalmente projetado para operar um sistema de tempo-real.

- Conjunto de instruções (simples ou complexas?).
- Modos de endereçamento (tempo de acesso a memória).

# Hardware para Sistemas de Tempo Real

## Itens a serem observados:

O hardware deve estar totalmente projetado para operar um sistema de tempo-real.

- Conjunto de instruções (simples ou complexas?).
- Modos de endereçamento (tempo de acesso a memória).
- Co-processadores (instruções mais rápidas e especializadas).

# Hardware para Sistemas de Tempo Real

## Objetivo do Projeto de Hardware para Sistemas de Tempo Real:

Utilização mais eficiente dos recursos de hardware.

Desempenho do software para satisfação das restrições temporais.

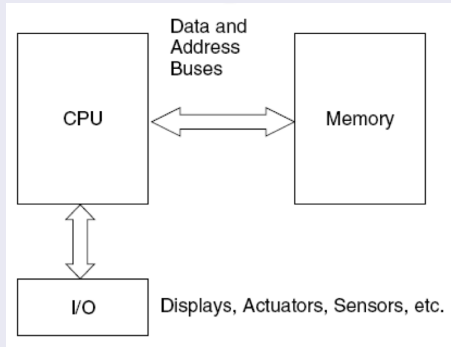
Previsibilidade [Coulouris et al. 2013]



# Arquitetura Básica de um Sistema Computacional

Arquitetura Básica de um Sistema Computacional consiste em:

CPU, Memória, Dispositivos de E/S interconectados através de um barramento, etc.



# Arquitetura com DMA

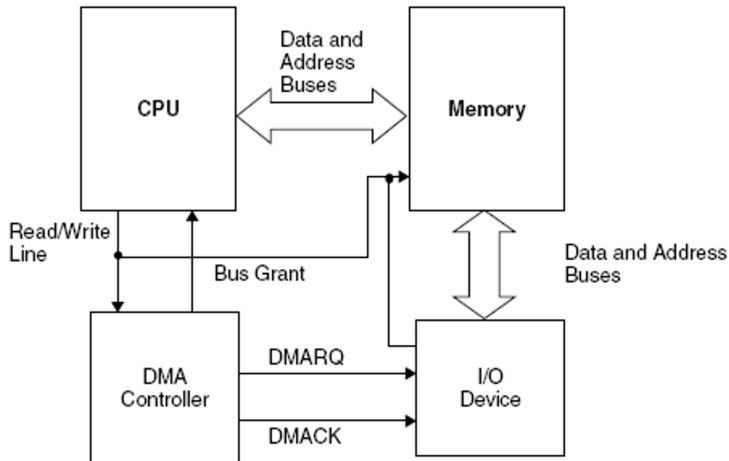
## Acesso direto a Memória (DMA):

Dispositivos lêem e escrevem dados diretamente na memória sem a intervenção da CPU. Requer um controlador de DMA (integrado ou não a CPU).

O Controlador de DMA evita colisões de dispositivos que requerem acesso a memória.

O dispositivo DMA insere imprevisibilidade no sistema.  
[TANENBAUM e STEEN 2007]

# Arquitetura com DMA



# Hardware para Sistemas de Tempo Real

O que é importante?

# Hardware para Sistemas de Tempo Real

## O que é importante?

- Velocidade de comunicação entre dispositivos (barramento – DMA - interfaceamento).

# Hardware para Sistemas de Tempo Real

## O que é importante?

- Velocidade de comunicação entre dispositivos (barramento – DMA - interfaceamento).
- Tecnologia utilizada na construção do processador (tipo de **pipeline**).

# Hardware para Sistemas de Tempo Real

## O que é importante?

- Velocidade de comunicação entre dispositivos (barramento – DMA - interfaceamento).
- Tecnologia utilizada na construção do processador (tipo de **pipeline**).
- Entender o funcionamento do hardware, pois ajuda a observar questões pertinentes ao tempo de resposta dos programas de Sistemas de Tempo Real.

# Pipeline

Um processador típico possui um pipeline com as seguintes atividades:



# Pipeline

Um processador típico possui um pipeline com as seguintes atividades:

- Busca de Instruções

# Pipeline

Um processador típico possui um pipeline com as seguintes atividades:

- Busca de Instruções
- Decodificação de operação

# Pipeline

Um processador típico possui um pipeline com as seguintes atividades:

- Busca de Instruções
- Decodificação de operação
- Execução

# Pipeline

Um processador típico possui um pipeline com as seguintes atividades:

- Busca de Instruções
- Decodificação de operação
- Execução
- Acesso à memória

# Pipeline

Um processador típico possui um pipeline com as seguintes atividades:

- Busca de Instruções
- Decodificação de operação
- Execução
- Acesso à memória
- Escrita de Retorno (Write-back)

# Pipeline

Em geral, cada instrução pode levar de 3 a 5 passos:

Busca, decodificação e execução são passos básicos.

Os demais dependem do tipo de instrução:

- Operações de acesso a memória (LOAD / STORE)
- Write-back (LOAD)

# CISC X RISC

## Diferença entre CISC X RISC:

A grande diferença entre as Arquiteturas **CISC(Complex Instruction Set Computer)** e **RISC (Reduced Instruction Set Computer)** é a Capacidade e o Modo de Processamento. Enquanto as arquiteturas CISC executam instruções complexas, as arquiteturas RISC executam instruções reduzidas.

# CISC X RISC

## Diferença entre CISC X RISC:

A grande diferença entre as Arquiteturas **CISC(Complex Instruction Set Computer)** e **RISC (Reduced Instruction Set Computer)** é a Capacidade e o Modo de Processamento. Enquanto as arquiteturas CISC executam instruções complexas, as arquiteturas RISC executam instruções reduzidas.

- CISC: Instruções complexas que exigem vários ciclos de relógio para serem executadas - suporta Instruções com formatos variados.
- RISC: Instruções simples executadas em um ciclo de relógio - suporta apenas Instruções com formatos fixos.



# Processador

Desempenho de um processador é verificado por:

- Velocidade de execução dos programas

# Processador

Desempenho de um processador é verificado por:

- Velocidade de execução dos programas
- Tempo de Resposta - Tempo decorrido entre o início e o final da execução do processo - envolve tempo de CPU, E/S, acesso a memória, etc.

# Processador

Desempenho de um processador é verificado por:

- Velocidade de execução dos programas
- Tempo de Resposta - Tempo decorrido entre o início e o final da execução do processo - envolve tempo de CPU, E/S, acesso a memória, etc.
- Número de programas que podem ser executados por vez (paralelismo)

# Conclusão

## Conclusão sobre o Projeto de Sistemas de Tempo Real:

- Rapidez de execução deve ser observada com o intuito de cumprir as restrições temporais.

# Conclusão

## Conclusão sobre o Projeto de Sistemas de Tempo Real:

- Rapidez de execução deve ser observada com o intuito de cumprir as restrições temporais.
- Arquiteturas RISC e CISC são úteis e usadas dependendo da necessidade do nosso sistema.

# Conclusão

## Conclusão sobre o Projeto de Sistemas de Tempo Real:

- Rapidez de execução deve ser observada com o intuito de cumprir as restrições temporais.
- Arquiteturas RISC e CISC são úteis e usadas dependendo da necessidade do nosso sistema.
- Sistemas de Tempo Real Críticos com restrições temporais apertadas necessitam de arquiteturas velozes.

# Conclusão

## Conclusão sobre o Projeto de Sistemas de Tempo Real:

- Rapidez de execução deve ser observada com o intuito de cumprir as restrições temporais.
- Arquiteturas RISC e CISC são úteis e usadas dependendo da necessidade do nosso sistema.
- Sistemas de Tempo Real Críticos com restrições temporais apertadas necessitam de arquiteturas velozes.
- Sistemas de Tempo Real Não-Críticos podem ser implementados em hardwares com certo nível de imprevisibilidade.

# Conclusão




## Conclusão sobre o Projeto de Sistemas de Tempo Real:

- O aumento da imprevisibilidade, normalmente, implica em diminuição de custos, por isso não devemos superdimensionar o sistema ou trabalhar com sistemas robustos para Sistemas de Tempo Real Não Críticos. Deve-se pensar no Custo X Benefício



# Referências

# Referências

-  BURNS, A.; WELLINGS, A. *Real-Time Systems and Programming Languages*. 1. ed. [S.I.]: Addison Wesley, 1997. v. 1.
-  COULOURIS, G. et al. *Sistemas Distribuídos: Conceitos e Projetos*. 5. ed. [S.I.]: Bookman, 2013. v. 1.
-  TANENBAUM, A.; STEEN, M. V. *Sistemas Distribuídos - Princípios e Paradigmas*. 2. ed. [S.I.]: Prentice Hall, 2007. v. 1.