



SISTEMAS DE I/O

Igor Monteiro Nunes

SUMARIO

- PAPEL DO SISTEMA OPERACIONAL
- SUBSISTEMA DE I/O DO KERNEL
- HARDWARE DE I/O
- SONDAGEM (POLLING)
- PROBLEMAS COM A SONDAGEM
- INTERRUPÇÕES
- ACESSO DIRETO À MEMÓRIA



PAPEL DO SISTEMA OPERACIONAL

- 
- GERENCIAMENTO DE I/O
 - HARDWARE DE I/O
 - SERVIÇOS DE I/O DO SO
 - STREAMS no UNIX System V
 - DESEMPENHO DE I/O



GERENCIAMENTO DE I/O

Garante que os **recursos** sejam utilizados de forma **eficiente** e **coordenada**, sem que as aplicações precisem lidar diretamente com os detalhes de **hardware**.



HARDWARE DE I/O

A **interface** entre o **sistema operacional** e o **hardware** impõe certas limitações e restrições, como a necessidade de **drivers** de dispositivos para facilitar a comunicação.



SERVIÇOS DE I/O DO SO

O sistema operacional fornece **serviços** específicos para operações de I/O, como **leitura** e **escrita** em arquivos, controle de dispositivos e comunicação de dados.



STREAMS no UNIX System V

Esse mecanismo do UNIX facilita a **construção** de sistemas modulares, em que os **componentes** de software que lidam com I/O (como drivers) podem ser conectados de **forma flexível** para atender às necessidades das aplicações.

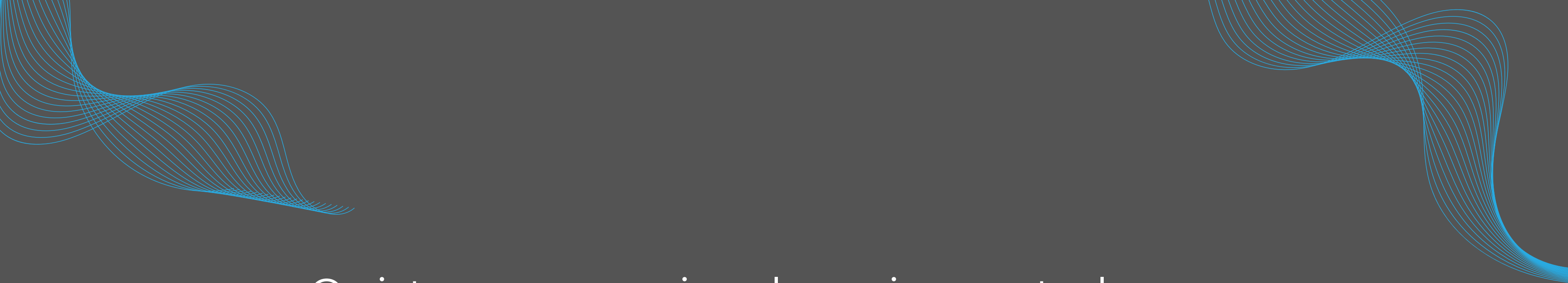


Desempenho de I/O


Melhorar o **desempenho** de operações de I/O é um dos **principais** objetivos do design de sistemas operacionais



SUBSISTEMA DE I/O DO KERNEL




O sistema operacional precisa controlar uma grande variedade de **dispositivos** de I/O, que diferem muito em **função** e **velocidade**. Para lidar com essa diversidade, ele usa o subsistema de I/O no **kernel**, que isola o restante do sistema da **complexidade** de gerenciar esses dispositivos.




Os **drivers** de dispositivos encapsulam os detalhes específicos de cada dispositivo, fornecendo uma interface **uniforme** para o subsistema de I/O do sistema operacional




HARDWARE DE I/O

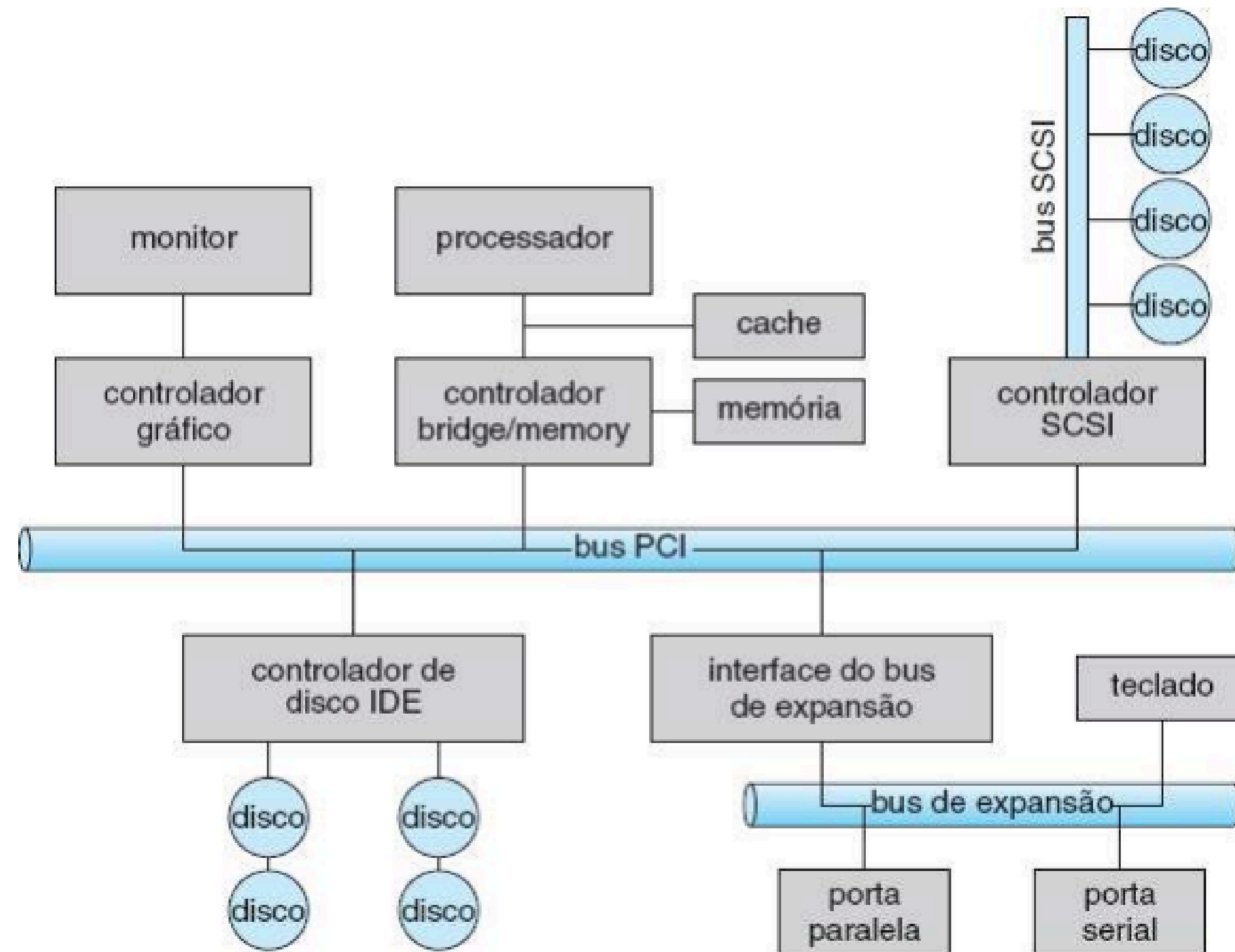


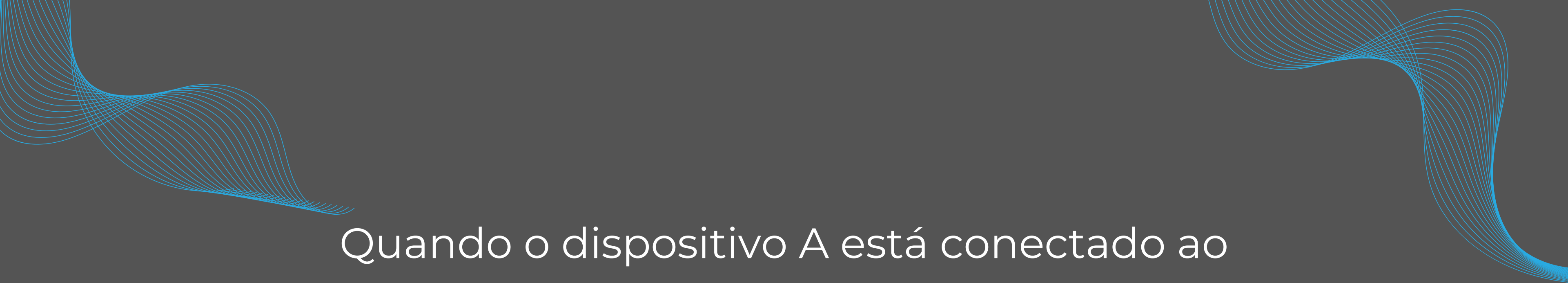
Um dispositivo de I/O se comunica com o computador por meio de **portas** (como portas seriais) ou por um **bus**, que é um conjunto de fios compartilhados com um protocolo para transmissão de mensagens



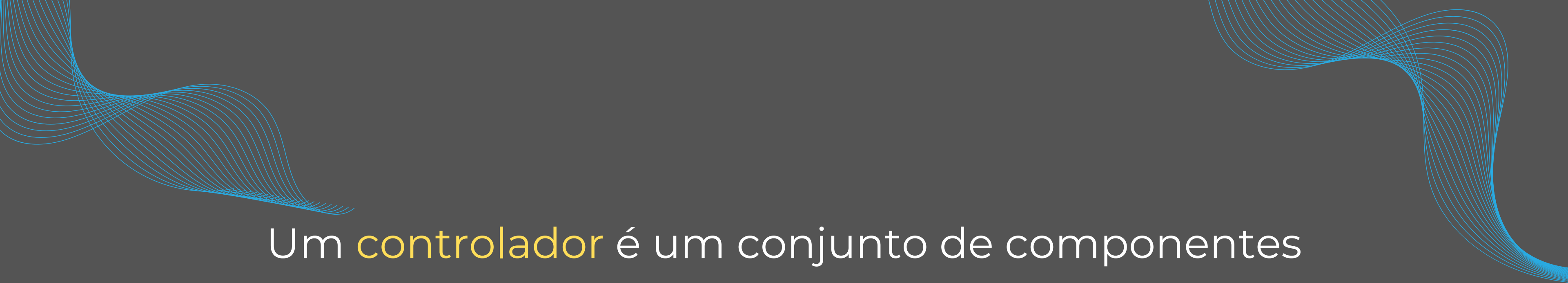
Os **buses** são amplamente utilizados na arquitetura de computadores e diferem em termos de **sinalização, velocidade, throughput** (capacidade de transmissão de dados) e **métodos de conexão**.

- 
- **PCI**: É um bus interno usado para conectar dispositivos de alta velocidade, como placas de vídeo, placas de som
 - **De expansão**: É utilizado para conectar dispositivos periféricos mais lentos, como teclados, mouses
 - **SCSI**: É usado principalmente para conectar dispositivos de armazenamento e alguns dispositivos de I/O, como discos rígidos







Quando o dispositivo A está conectado ao dispositivo B por um cabo, e o dispositivo B, por sua vez, está conectado ao dispositivo C, que então se conecta a uma porta no computador, essa configuração é chamada de **cadeia margarida**. Normalmente, uma cadeia margarida funciona da mesma maneira que um **bus**, permitindo que os dispositivos compartilhem a conexão de dados.



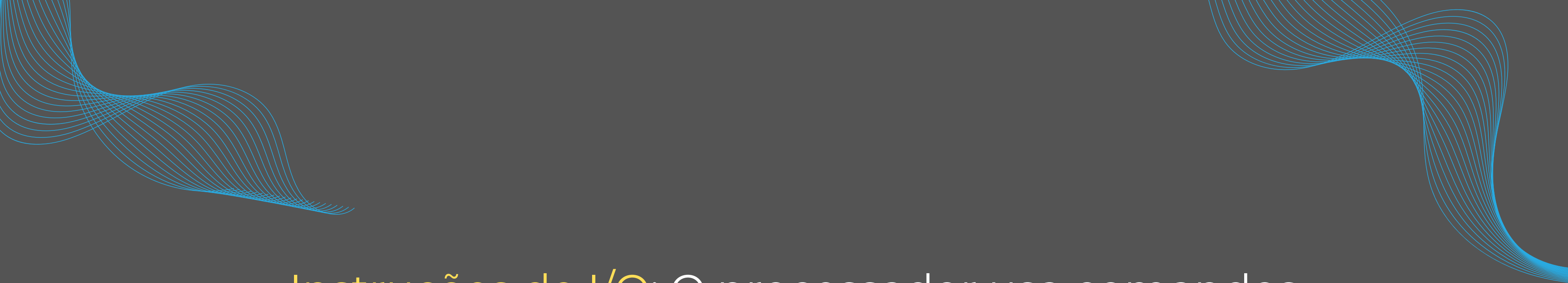
Um **controlador** é um conjunto de componentes eletrônicos que gerencia a operação de uma porta, bus ou dispositivo. Existem controladores **simples**, como os de porta serial, que controlam sinais básicos em fios, e controladores **mais complexos**, como o controlador SCSI, que requer uma placa separada com processador, microcódigo e memória para lidar com o protocolo avançado do SCSI.




Como o processador pode fornecer comandos e dados a um controlador para realizar uma transferência de I/O?




Um controlador tem **registradores** que armazenam dados e sinais de controle. O processador se comunica com o controlador lendo e escrevendo bits nesses registradores.


- 
- . **Instruções de I/O:** O processador usa comandos especiais para transferir dados para um endereço de porta de I/O, que envia sinais ao controlador e movimenta os bits.
 - . **I/O mapeado para a memória:** Os registradores do controlador são mapeados no espaço de endereçamento da memória do processador.

intervalo de endereços de I/O (hexadecimal)	dispositivo
000–00F	controlador de DMA
020–021	controlador de interrupções
040–043	timer
200–20F	controlador de jogos
2F8–2FF	porta serial (secundária)
320–32F	controlador de disco rígido
378–37F	porta paralela
3D0–3DF	controlador gráfico
3F0–3F7	controlador de drive de disquete
3F8–3FF	porta serial (primária)




Uma porta de I/O consiste, tipicamente, em **quatro** registradores, chamados de registradores de status, controle, dados de entrada e dados de saída

- 
- **Registrador de dados de entrada:** contém os dados que o dispositivo envia ao hospedeiro (a CPU)
 - **Registrador de dados de saída:** O hospedeiro grava neste registrador para enviar dados ao dispositivo

- 
- **Registrador de status:** Esse registrador armazena informações (contem bits) sobre o estado do dispositivo (concluído, disponível) que podem ser lidas pelo hospedeiro
 - **Registrador de controle:** O hospedeiro grava neste registrador para enviar comandos e alterar o comportamento do dispositivo. Ele é usado para iniciar operações, como ligar ou desligar um dispositivo

The image features a dark gray background with decorative wavy lines in the top corners. The lines are composed of many thin, overlapping curves in a light blue or cyan color, creating a sense of motion and depth. The lines in the top-left corner curve downwards and to the right, while the lines in the top-right corner curve downwards and to the left.

Sondagem (Polling)




O **aperto de mãos (Handshake)** é um mecanismo de sincronização entre o controlador e o hospedeiro para garantir que as operações de leitura ou escrita ocorram na ordem correta e sem conflitos




Para coordenar as ações, o hospedeiro e o controlador usam **dois bits** principais:


- **Bit busy (ocupado)**: usado pelo controlador. Ele é ativado (ligado) quando o controlador está ocupado e desligado quando o controlador está pronto para aceitar outro comando.

- 
- **Bit command-ready (comando pronto):** usado pelo hospedeiro. Ele é ativado quando o hospedeiro está pronto para enviar um comando ao controlador.

Etapas Handshake


- **Hospedeiro verifica se o controlador está pronto:**
O hospedeiro lê repetidamente o bit busy no registrador de status até que ele seja desligado
- **Hospedeiro envia um comando:** O hospedeiro ativa o bit write no registrador de comando e grava um byte no registrador de saída e , ativa o bit command-ready

- 
- **Controlador reconhece o comando:** O controlador percebe que o bit command-ready foi ativado e liga o bit busy para indicar que está ocupado processando o comando
 - **Controlador executa a operação de I/O:** O controlador lê o registrador command para verificar o tipo de comando e lê o byte do registrador de saída e executa a operação de I/O no dispositivo

- 
- Finalização do ciclo:
 - . Desativa o bit command-ready
 - . Desliga o bit busy
 - . Se o I/O foi bem-sucedido, desliga o bit de erro



Problemas com a Sondagem



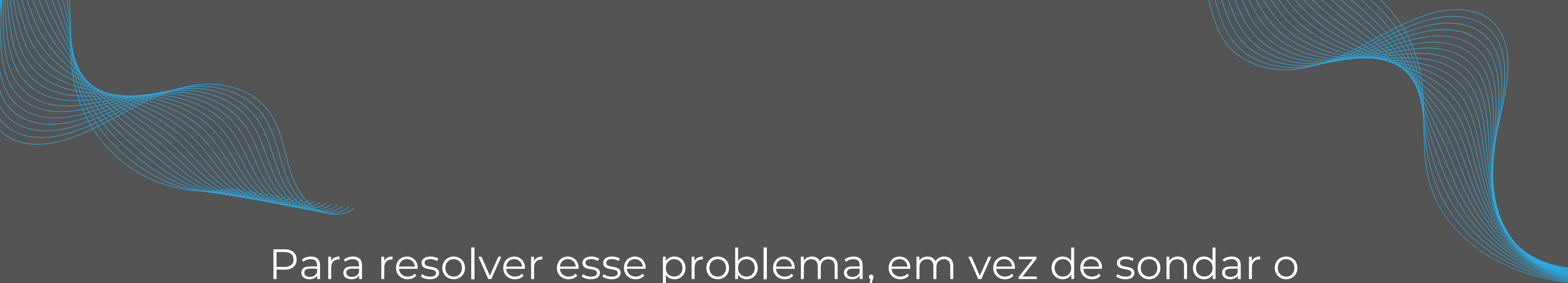
No passo 1, o hospedeiro pode entrar em um **loop de sondagem (polling)**, verificando repetidamente o status do bit busy até o controlador ficar pronto.

Se o controlador for rápido, isso funciona bem. Porém, se o controlador demorar para processar o comando, o hospedeiro fica preso nesse loop, desperdiçando recursos.

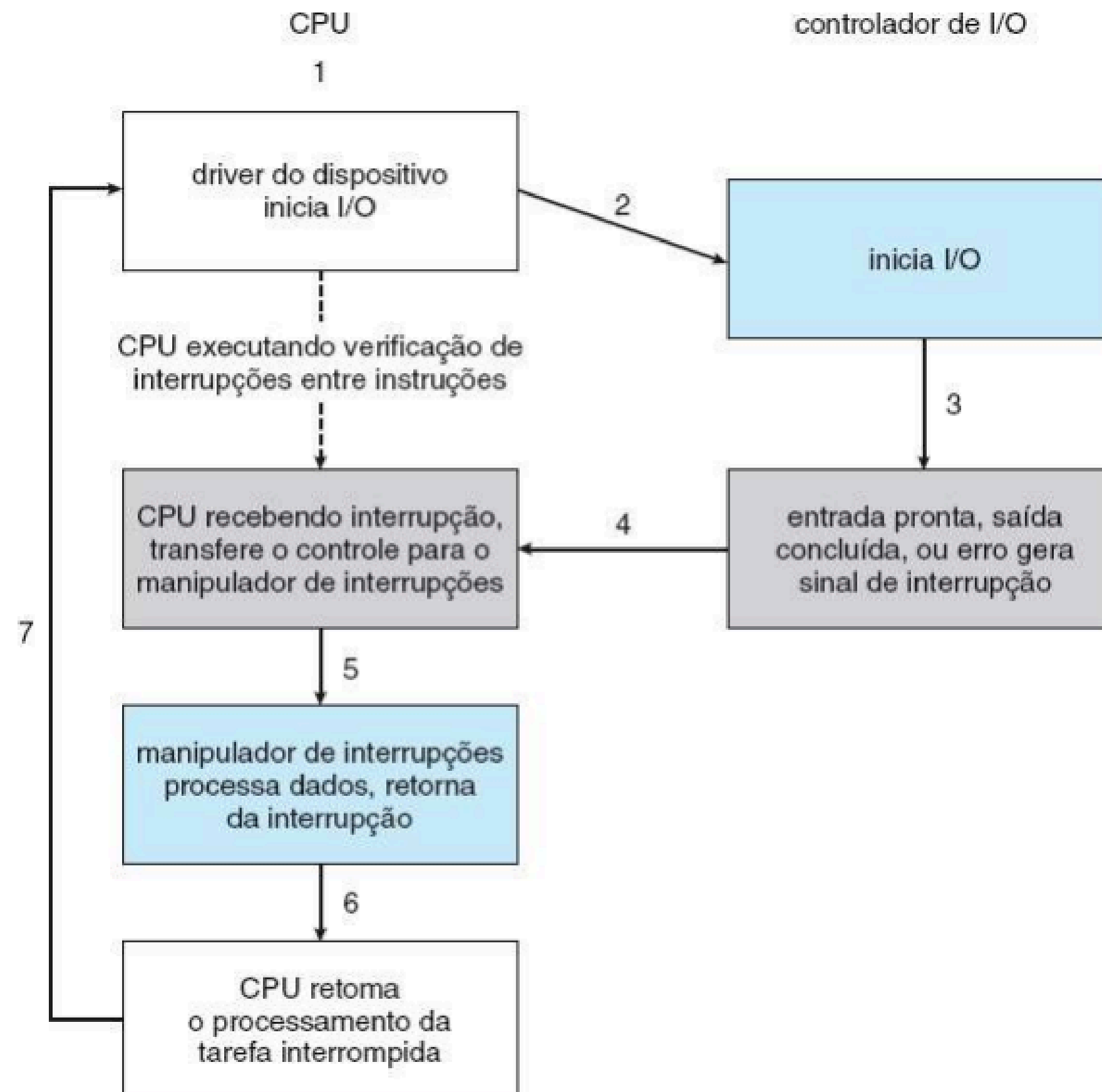


Torna-se **ineficiente** pois

- A CPU gasta **tempo** verificando o status, sem realizar outras tarefas úteis
- O hospedeiro pode perder **dados críticos**, como em dispositivos de alta velocidade



Para resolver esse problema, em vez de sondar o controlador repetidamente, o mecanismo de **interrupções** permite que o controlador notifique o hospedeiro assim que estiver pronto. Isso torna o processo mais eficiente, liberando o hospedeiro para executar **outras tarefas** enquanto aguarda a notificação.






INTERRUPÇÕES



Linhas de Solicitação de Interrupção

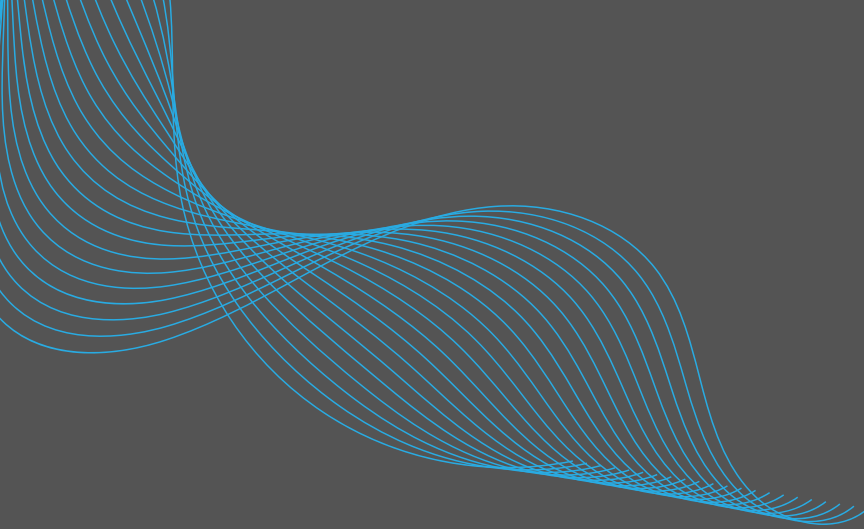
O hardware da CPU possui um canal físico ou lógico chamado linha de solicitação de interrupção que a CPU examina após executar cada instrução de I/O

- 
- **Interrupção não mascarável:** Usada para eventos críticos, como erros de memória irre recuperáveis, que não podem ser ignorados.
 - **Interrupção mascarável:** Usada para interrupções geradas por dispositivos e pode ser temporariamente desativada (mascarada) pela CPU em momentos críticos



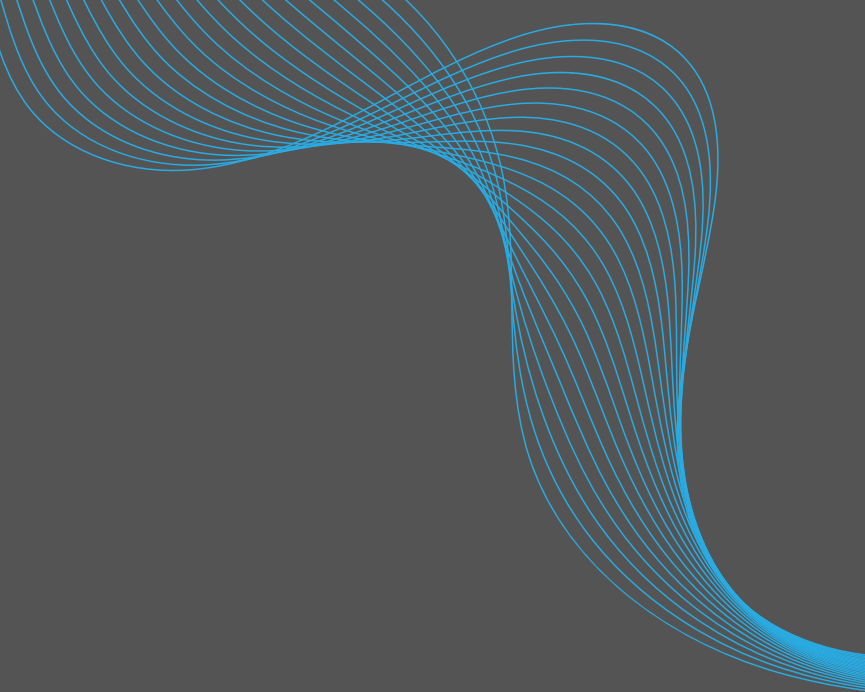
Vetor de Interrupções

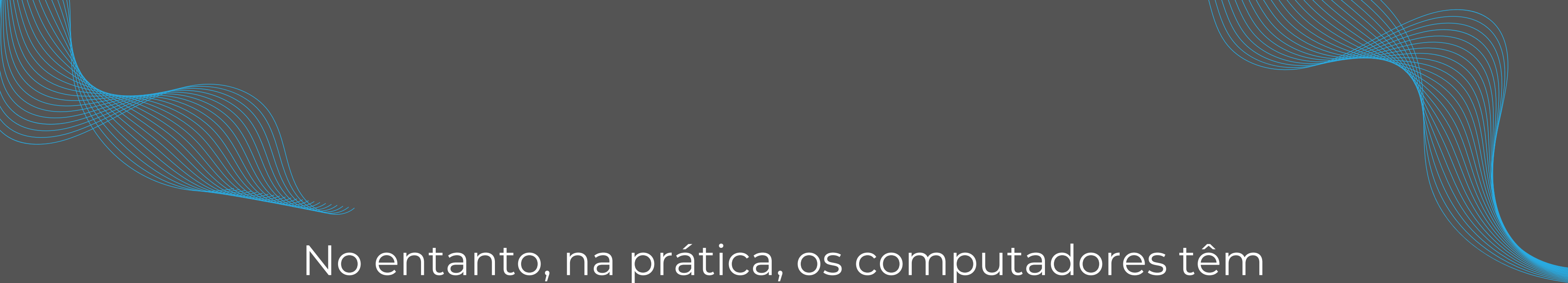
O sistema de interrupções utiliza um **vetor de interrupções**, que é uma tabela de endereços de memória apontando para as **rotinas de tratamento de interrupções específicas**. Quando uma interrupção é gerada, o sistema não precisa verificar cada dispositivo para descobrir a causa; ele vai diretamente ao manipulador apropriado no vetor



número do vetor	descrição
00	erro de divisão
01	exceção de depuração
02	interrupção nula

03	ponto de interrupção
04	estouro INTO detectado
05	exceção de intervalo limite
06	código de operação inválido
07	dispositivo não disponível
08	erro duplo
09	sobrecarga do segmento do coprocessador (reservado)
10	segmento de estado de tarefa inválido
11	segmento não presente
12	erro de pilha
13	proteção geral
14	erro de página
15	(reservado para a Intel, não use)
16	erro de ponto flutuante
17	verificação de alinhamento
18	verificação de máquina
19-31	(reservado para a Intel, não use)
32-255	interrupções mascaráveis






No entanto, na prática, os computadores têm mais dispositivos (e, portanto, mais manipuladores de interrupções) do que elementos de endereço no vetor de interrupções. Então, como existem mais dispositivos do que entradas no vetor, um **encadeamento de interrupções** é usado, onde cada entrada no vetor aponta para uma lista de manipuladores que são chamados um por um até encontrar o correto.



Níveis de Prioridade

O sistema de níveis de prioridades de interrupções permite que a CPU gerencie **múltiplas interrupções** de diferentes dispositivos e determine quais devem ser tratadas com mais urgência.



Além de gerenciar I/O de dispositivos, os sistemas operacionais utilizam o mecanismo de interrupção para:

- Tratamento de exceções: Como tentativas de acessar áreas protegidas de memória
- Gerenciamento de memória virtual: onde falhas de página (page faults) são tratadas por interrupções



Acesso Direto à Memória




PIO (Programmed I/O)

No PIO, a CPU é responsável por verificar os bits de status de um dispositivo e transferir os dados **byte a byte** para o registrador do controlador. Isso pode ser **ineficiente**, pois a CPU fica sobrecarregada com tarefas repetitivas que poderiam ser automatizadas.



DMA (Direct Memory Access)

Para evitar sobrecarregar a CPU com I/O programado, computadores modernos utilizam **controladores de DMA**. Esses controladores são processadores especializados que assumem a tarefa de **transferência de dados**, permitindo que a CPU continue com outras atividades.




O funcionamento básico do DMA envolve o **bloco de comando DMA**, que contém informações como a origem e o destino da transferência de dados, além da **quantidade de bytes** a serem transferidos




Operação DMA

O DMA se comunica com o controlador do dispositivo por meio de sinais chamados **solicitação de DMA** (DMA request) e **reconhecimento de DMA** (DMA acknowledge)



Quando o dispositivo tem dados prontos para serem transferidos, ele envia um sinal de **solicitação de DMA**. O controlador de DMA, por sua vez, assume o controle do bus de memória, realiza a transferência dos dados e envia um sinal de **reconhecimento ao dispositivo**, confirmando a conclusão da operação.




Ao final da transferência, o controlador de DMA **interrompe a CPU** para informá-la que os dados foram transferidos.



Vantagem do DMA

Apesar do roubo de ciclos (que ocorre quando o DMA temporariamente bloqueia o acesso da CPU à memória), o DMA melhora o desempenho geral do sistema, já que **libera a CPU** para realizar outras tarefas enquanto o controlador cuida da **movimentação dos dados**.



Em sistemas operacionais protegidos, o acesso direto a dispositivos é controlado pelo **kernel**, para evitar **falhas de segurança** e o **mau uso dos controladores de dispositivos**, o que poderia derrubar o sistema

Em sistemas sem proteção de memória, processos **podem acessar os controladores diretamente**, o que pode aumentar o desempenho, mas compromete a segurança e a estabilidade do sistema.



Bibliografia

SILBERSCHATZ, A.; GALVIN, P. B.; GREG, G.
Fundamentos de Sistemas Operacionais. 9º
ed. Rio de Janeiro: LTC, 2015