

A photograph showing a portion of a desk. On the desk is a brown leather office chair, a white desk surface, a black keyboard, and a black mouse. In the background, there are window blinds and a grey wall.

# Gerenciamento de I/O e Interrupções

---

## Parte 2 - Variantes de Processamento de I/O

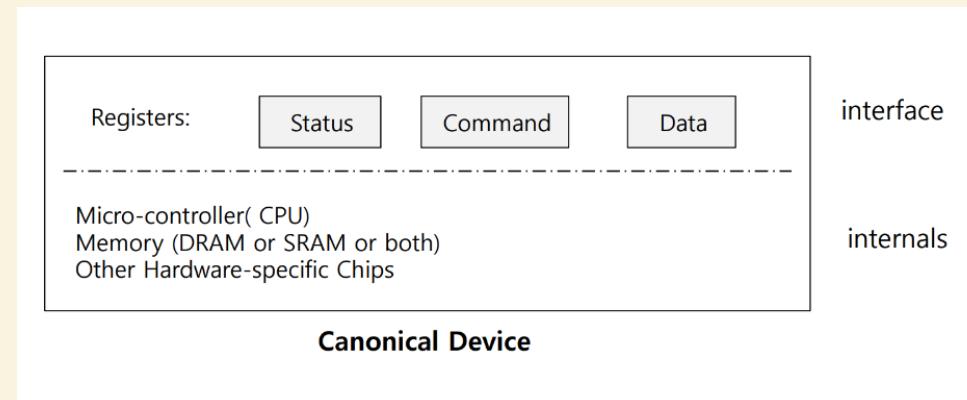
Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

# I/O Programado

---

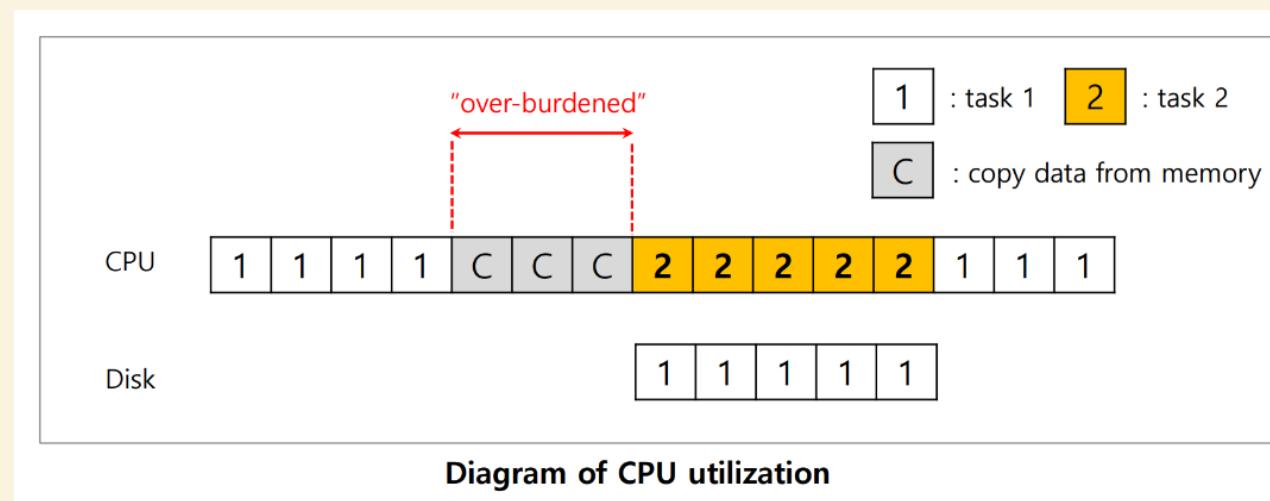
- Quando há envolvimento da CPU na transferência de dados.
- **Polling:** SO espera o dispositivo estar pronto > envia dados para o registrador de dados do dispositivo > SO escreve comando no registrador de comando do dispositivo > SO espera por resposta do dispositivo (loop/busy wait).
- **Interrupção:** SO faz requisição ao dispositivo > Mudança de contexto > Dispositivo aciona interrupção, causando a CPU a utilizar um interrupt handler.



Fonte da Imagem: [Operating Systems: Three Easy Pieces](#)

# Problema com I/O Programado

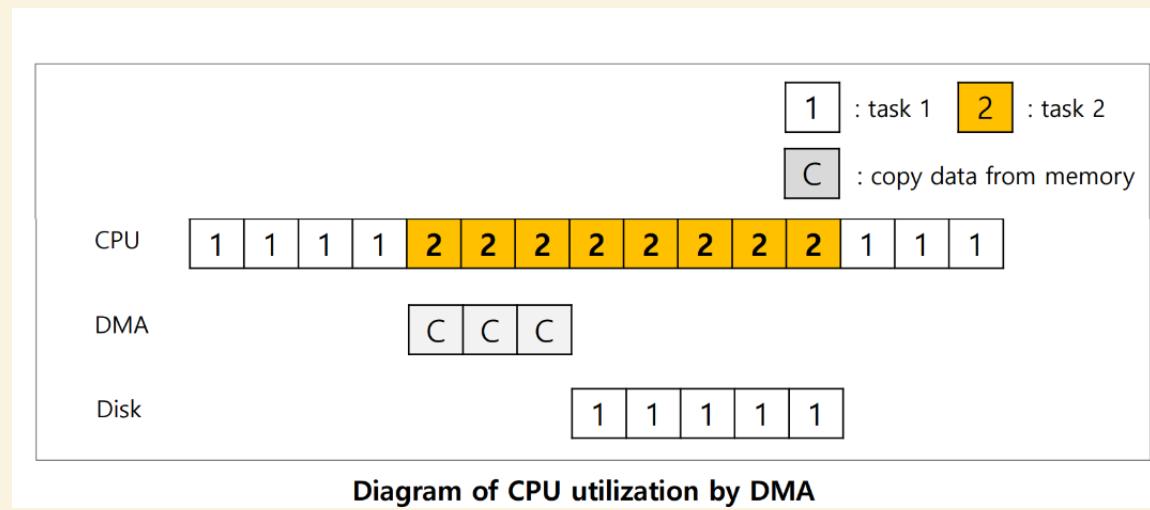
- Para um dispositivo que faz grandes transferências, como, por exemplo, um drive de disco, a CPU é "sobre carregada" com tarefas triviais:
  - Verificar bits de status
  - Alimentar dados, byte a byte
- Parece desperdício usar CPU nessas. Melhor utilizá-la executando outros processos.



Fonte da Imagem: Anotações de JungJae Lee

# Acesso Direto à Memória (DMA)

- Utilizado para grandes transferências de dados
- Dispensa CPU na movimentação de dados
- SO configura o controlador de DMA:
  - Endereços fonte/destino, modo de leitura ou escrita
  - Quantos dados copiar, contagem de bytes



Fonte da Imagem: [Anotações de JungJae Lee](#)

# Acesso Direto à Memória (DMA)

- Quando a transferência inteira termina, o controlador de DMA interrompe a CPU (**interrupção**).
- Quando o controlador de DMA se apropria do bus da memória, a CPU é momentaneamente impedida de acessar a memória principal, mas ainda pode acessar caches primário e secundário.
- Roubo de ciclos da CPU, mas mais eficiente

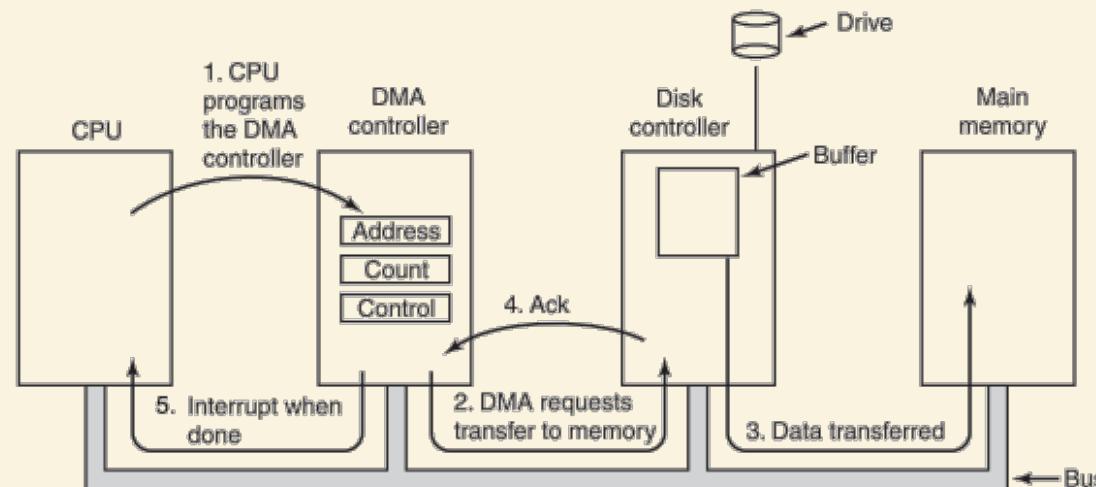
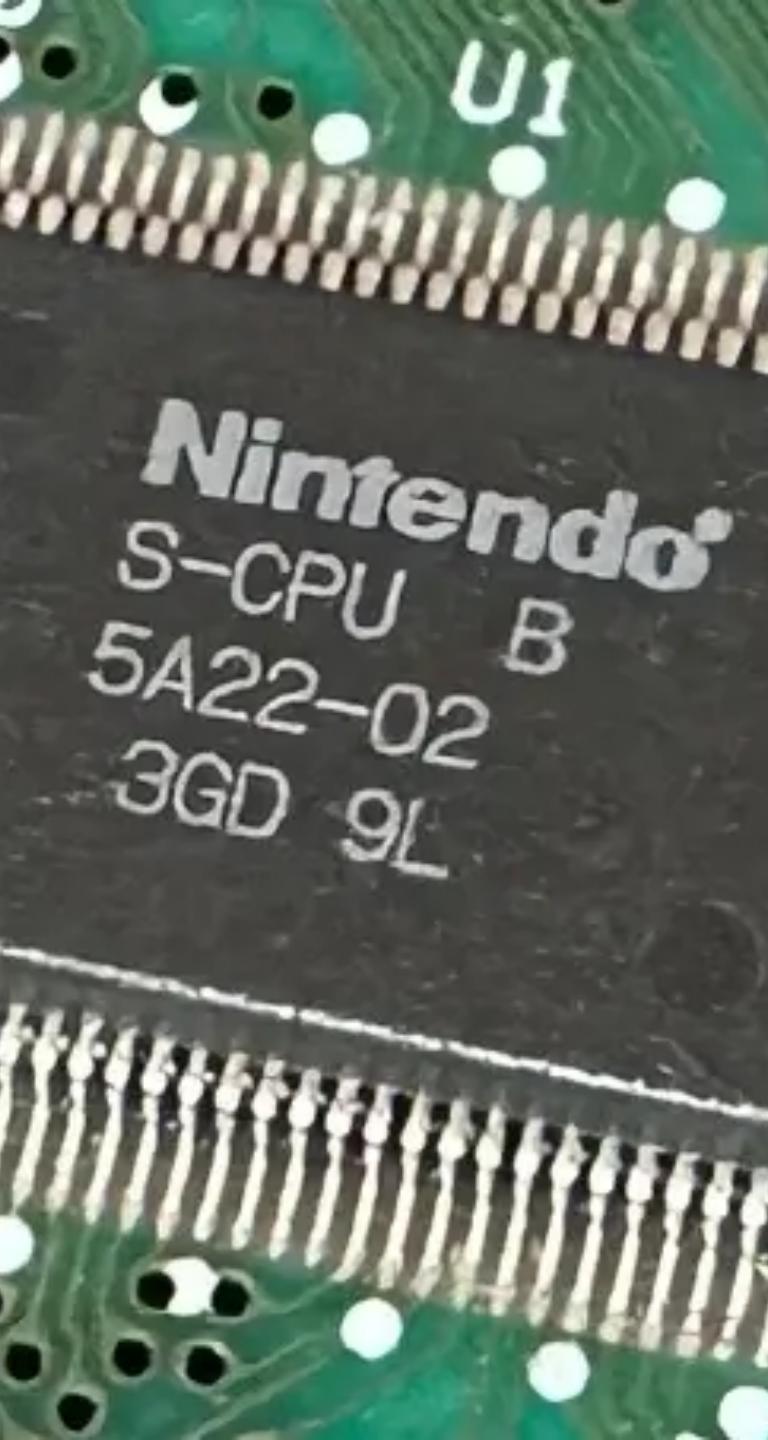


Figure 5-4. Operation of a DMA transfer.

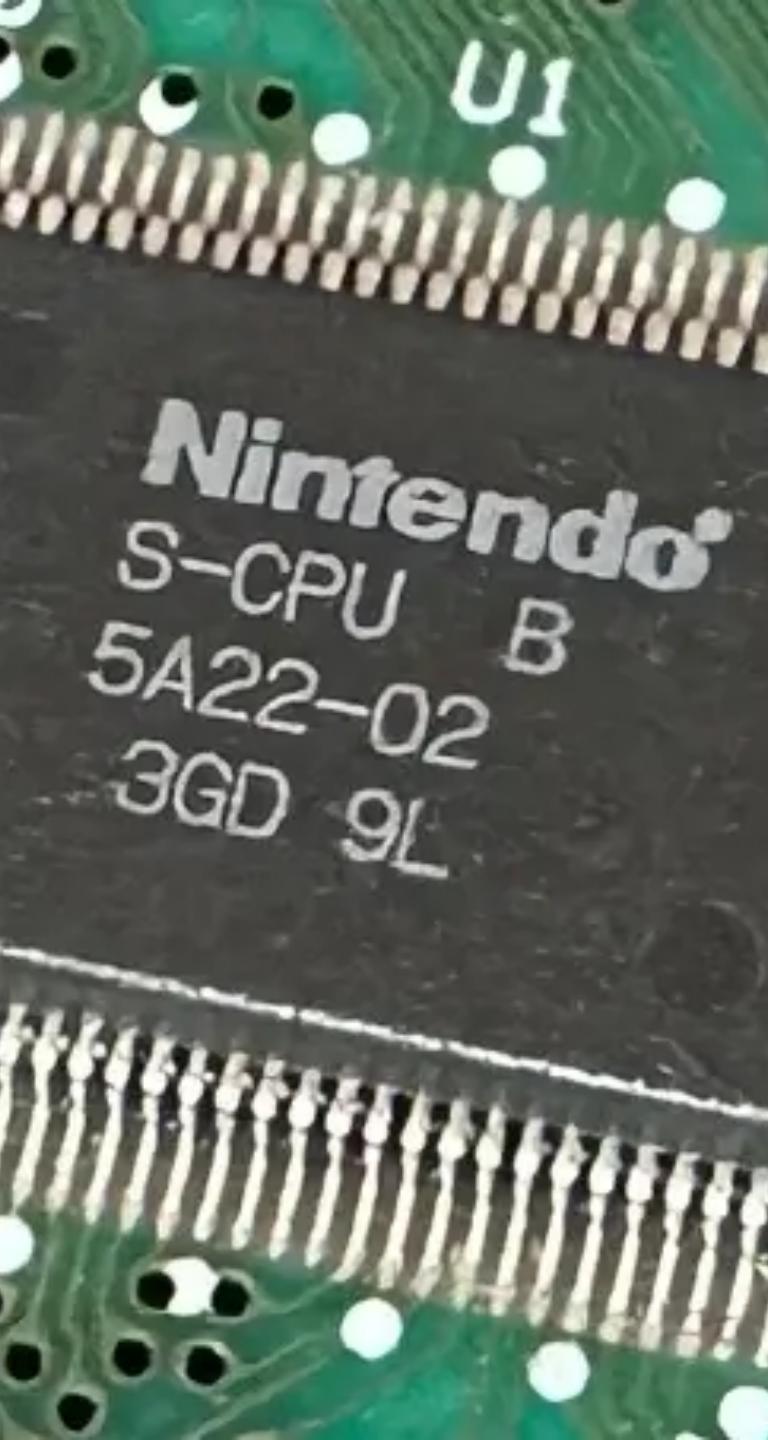
Fonte da Imagem: [Modern Operating Systems, Principles of I/O Hardware](#)



## DMA no Super Nintendo (1990)

"The second challenge was to increment its data bandwidth. Hence, two exclusive DMAs (Direct Memory Access) were added to move data around without the intervention of the CPU (resulting in faster speeds)." - Rodrigo Copetti em [Super Nintendo Architecture - A practical analysis](#) (Fonte da imagem).

Horizontal DMA (HDMA) no SNES: Evita interromper a CPU em longos intervalos, mas as transferências era limitadas em 4 bytes por [scanline](#) (enquanto CRT se prepara para exibir a próxima linha).

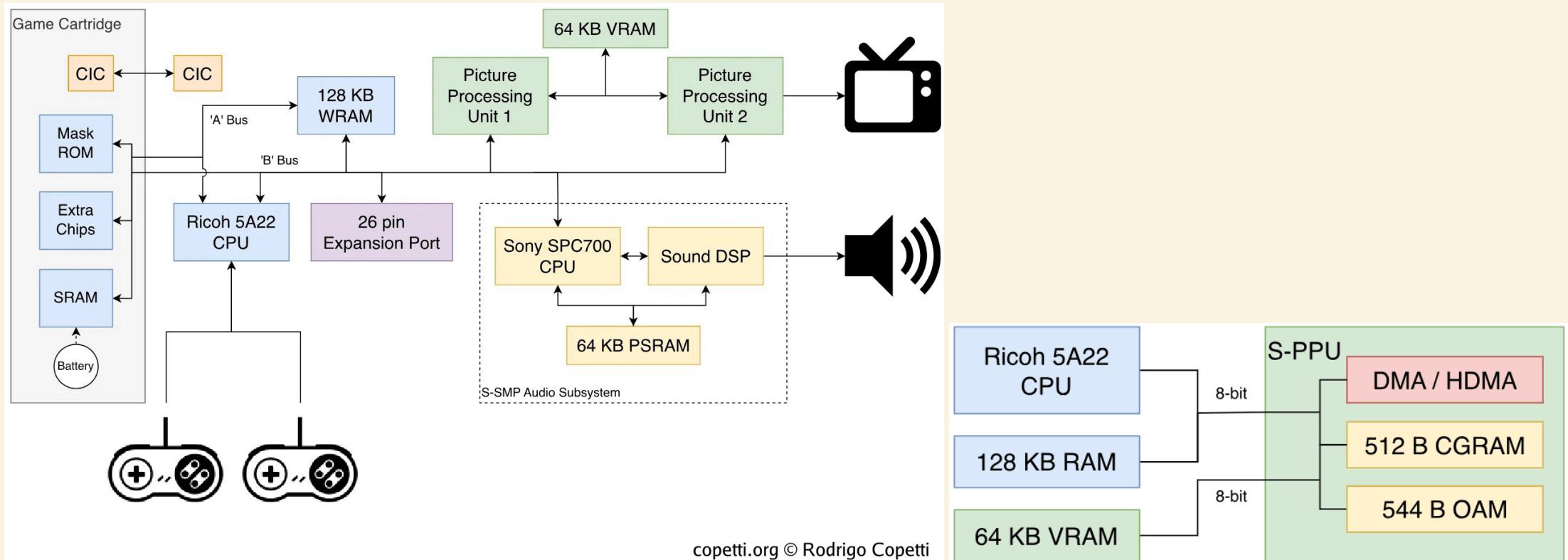


## DMA no Super Nintendo (1990)



"Game cartridges are no longer wired to the PPU so tiles will have to be copied to VRAM first (just like Sega's Mega Drive). Here is where the DMA unit comes in very handy. (...) games can now update tiles, colours and registers without waiting for the whole frame to be drawn." - Rodrigo Copetti em [Super Nintendo Architecture - A practical analysis.](#) (Fonte das imagens)

# DMA no Super Nintendo (1990)



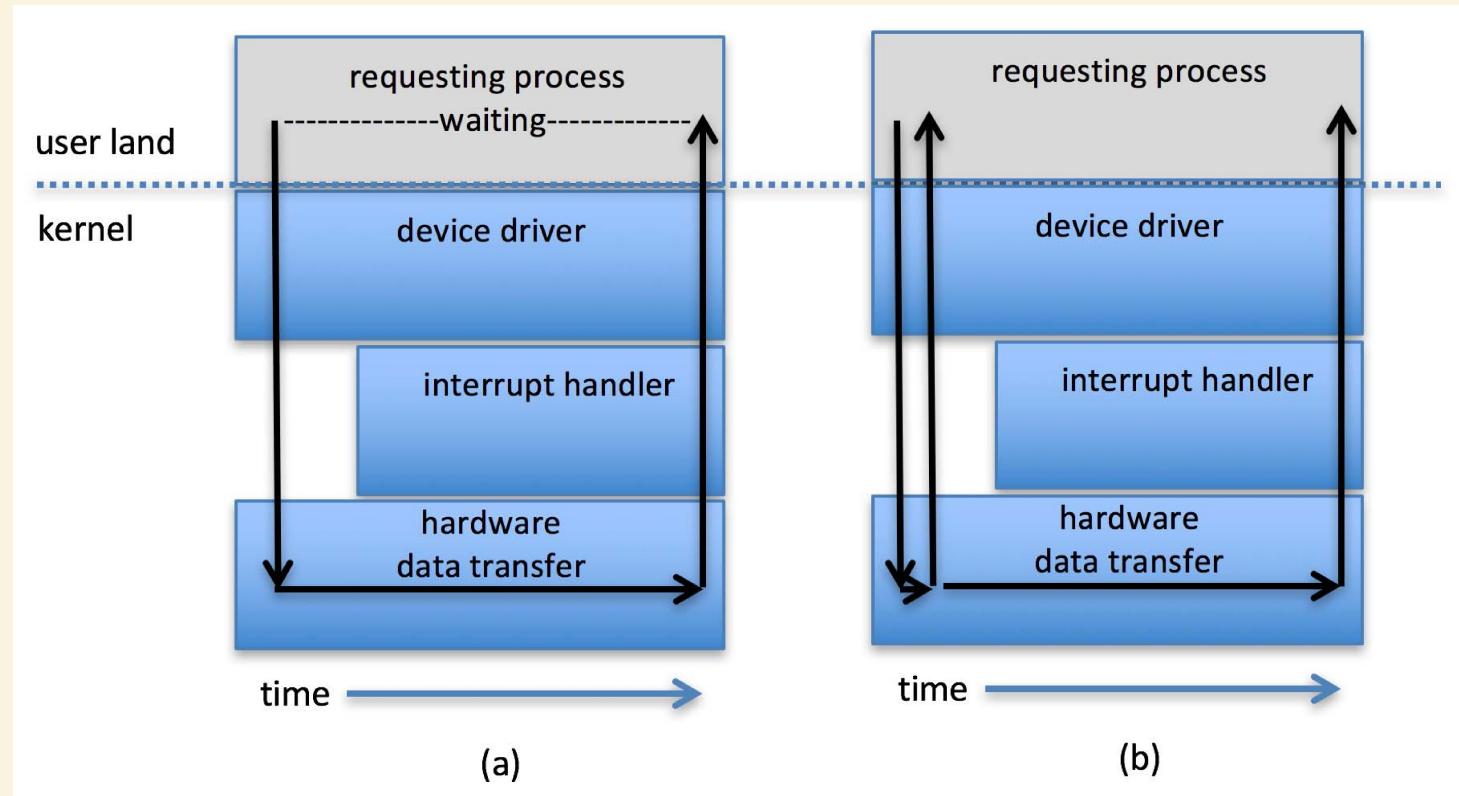
Fonte da Imagem: [Super Nintendo Architecture - A practical analysis by Rodrigo Copetti](#)

# I/O Não Bloqueante e Assíncrono

---

- Aplicação pode realizar chamada de sistema de diferentes tipos.
- **Bloqueante:** processo suspenso até o fim da operação.
  - Código fácil de usar e de entender
  - Usar threads para executar chamadas bloqueantes
- **Não Bloqueante:** retorna imediatamente com dados disponíveis.
- Exemplo: Interface de usuário
- **Assíncrono:** processo continua enquanto o I/O ocorre, sinalizado depois.

# Métodos de I/O

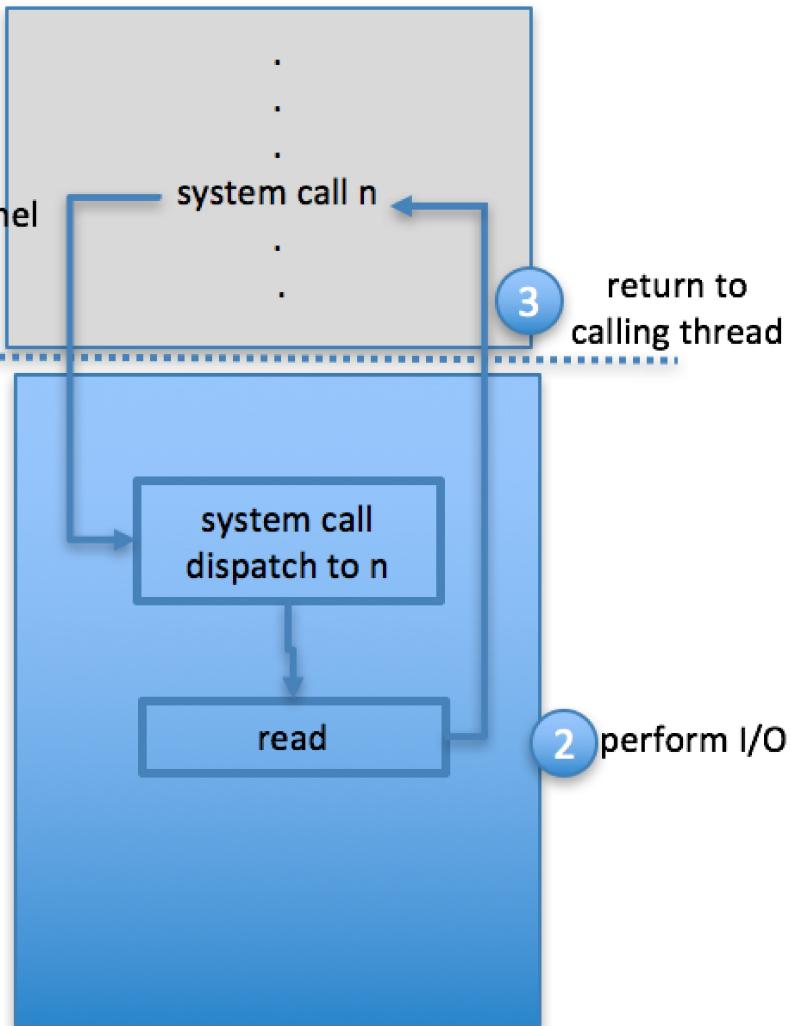


Dois métodos de I/O: (a) síncrono e (b) assíncrono. Fonte da Imagem: A. Silberschatz et. al, *Operating Systems Concepts*, capítulo 13.

# Subsistema de I/O no Kernel

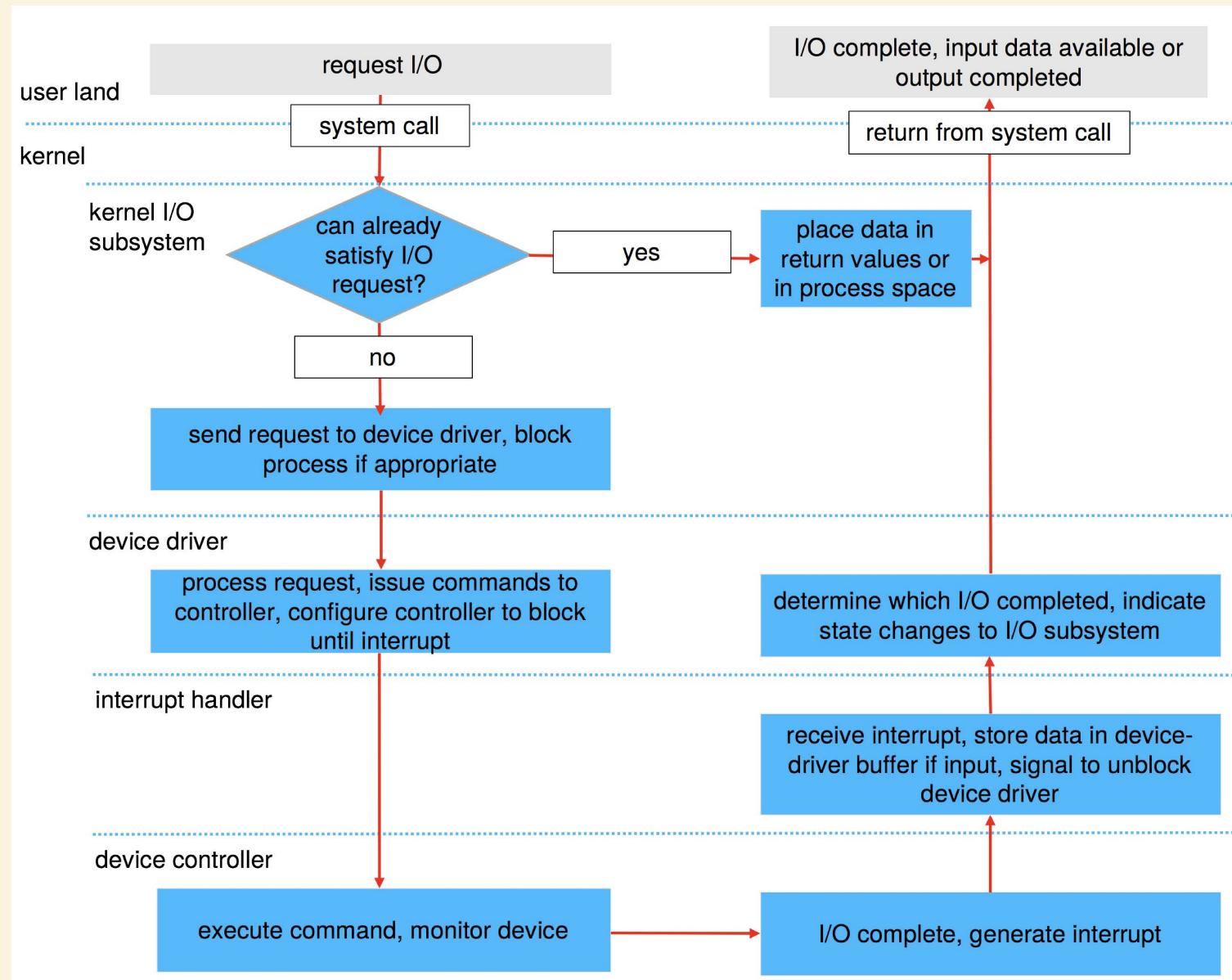
---

- Agendamento de requisições de I/O
- Bufferização para lidar com velocidades/tamanhos distintos
- Caching para performance
- Spooling para dispositivos exclusivos
- Reserva de dispositivos para acesso exclusivo



## Proteção de I/O

- Instruções de I/O são privilegiadas
- Acesso apenas por chamadas de sistema
- Endereços de dispositivos mapeados em memória protegidos
- Na imagem: Chamadas de Sistema para Operações de I/O.
  - Fonte: Silberschatz et. al, Operating Systems Concepts, cap 13.



Ciclo de vida de uma requisição de I/O. Fonte da Imagem: A. Silberschatz et. al, *Operating Systems Concepts*, capítulo 13.

# Performance em I/O

---

- I/O é fator crítico no desempenho do sistema
- Envolve:
  - Execução de drivers
  - Trocas de contexto
  - Cópias de dados
  - Tráfego de rede

# Melhorando a Performance

---

- Reduzir trocas de contexto
- Reduzir cópias de dados
- Minimizar interrupções usando transferências grandes
- Usar DMA
- Balancear CPU, memória, barramento e I/O
- Migrar processos de modo usuário para threads de kernel

# Interrupt Livelock

---

- Livelock: Situação em que as computações ocorrem, mas (quase) nenhum progresso é feito.
  - O tempo de computação é majoritariamente desperdiçado com sobrecarga.
- Livelock por Interrupção (Interrupt Livelock): As interrupções chegam tão rápido que não podem mais ser processadas.
  - Isso também deixa pouco tempo de CPU disponível para outras tarefas.
  - Troca de contexto, poluição do cache.
  - Nada útil acontece mais.
- Prevenir por uma combinação de polling e interrupções. Ex: [Linux NAPI](#).

# Conclusão

---

- I/O envolve **buses, controladores e dispositivos físicos**.
- A movimentação de dados é feita por **I/O programado** (CPU) ou **DMA** (acesso direto à memória).
- **Drivers de dispositivos** no kernel controlam o hardware.
- Aplicações interagem com I/O por meio de chamadas de sistema para:
  - Dispositivos de bloco e de caractere
  - Arquivos mapeados em memória
  - Sockets e timers
- Chamadas de I/O são caras em termos de consumo da CPU devido ao **overhead de camadas de software, mudança de contexto, sinais e cópias de dados** entre kernel e espaço do usuário.

# Leitura adicional

---

- Capítulo 5 do livro **Sistemas Operacionais Modernos**, de A. TANENBAUM
- Capítulo 13 do livro: **Operating Systems Concepts**, de A. Silberchatz et. al.

# Conteúdo Adicional

---

- [Creating an Operating System for the NES](#) -  [YouTube](#)
- [Linux Running On An NES?](#) -  [YouTube](#)
- [SNES-OS](#) - [GitHub](#)
- [Interrupts and I/O - Part 2](#) por [Jens Lechtenbörger](#) - Material de aula do curso de IT

System na Universidade de Münster  Universität  
Münster

# Dúvidas e Discussão

---