FACULDADE DE TECNOLOGIA SENAC RJ

ANÁLISE E DESENVOLVIMENTO DE SOFTWARE

SISTEMAS OPERACIONAIS

ENTRADA E SAÍDA - SOFTWARE

Conteúdo

1. Software de Entrada/saída

2. Drivers

3. Estratégias de interação
   - Entrada/saída por programa
   - Entrada/saída por eventos
   - Entrada/saída por DMA

4. Tratamento de interrupções

* referência:
Sistemas Operacionais – Conceitos e Mecanismos, Carlos Maziero, UFPR.

Software de entrada/saída

Software que interage com os dispositivos:

- *Drivers* ou pilotos
- Abstrações de baixo nível (sockets, blocos, etc)

Grande diversidade de dispositivos:

- Muitos dispositivos distintos = muitos *drivers*
- 60% do código fonte do Linux kernel são *device-drivers*

Estrutura geral do núcleo

Drivers

- Interagem com os dispositivos
- Acessam portas de E/S e tratam interrupções

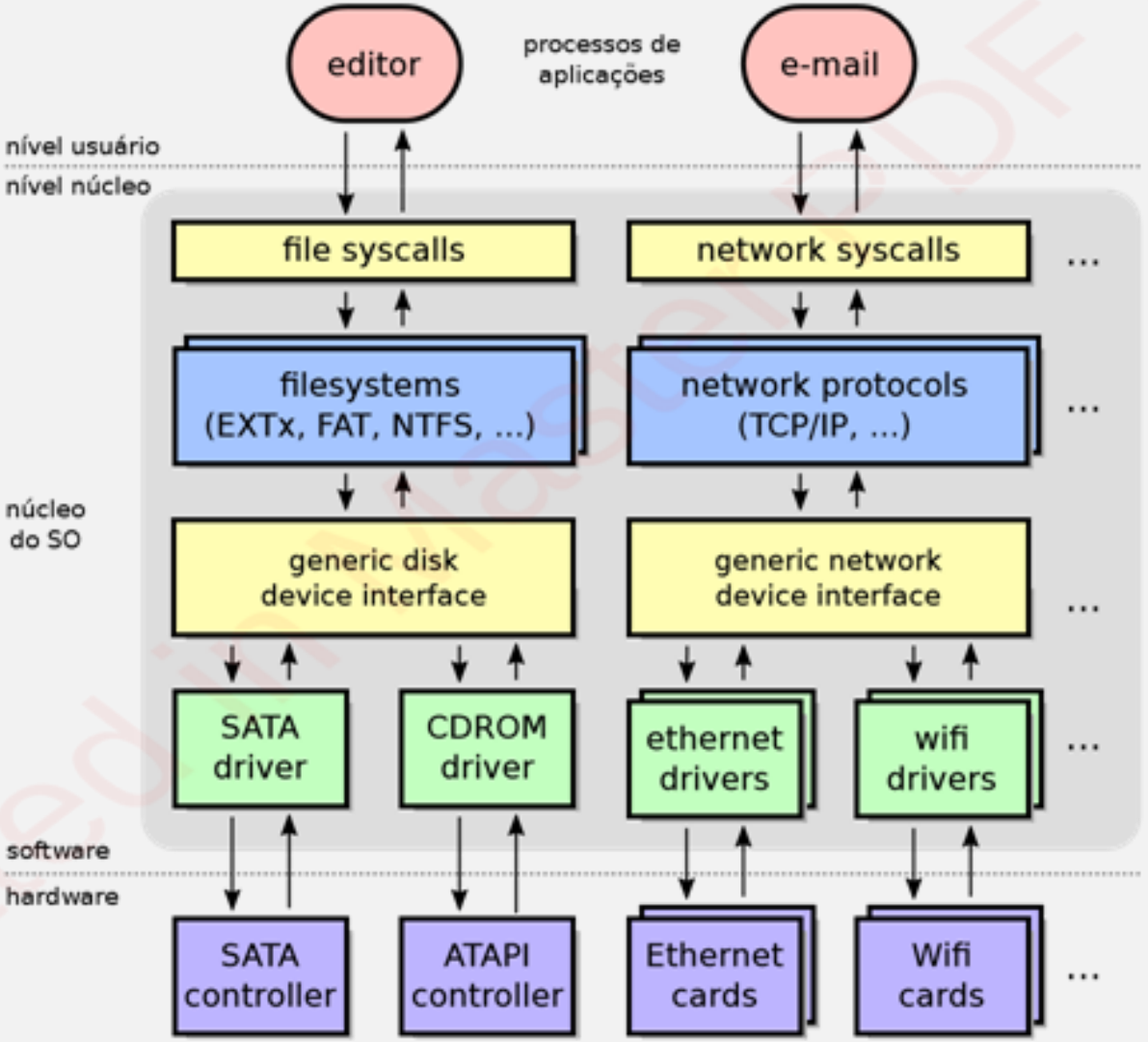
Dispositivos genéricos

- Visão genérica de dispositivos similares (discos)
  - Discos: vetores de blocos de dados
  - Interfaces de rede (especificação NDIS)

Abstrações: Sistemas de arquivos, protocolos de rede

Chamadas de sistema: interface oferecida aos processos

Estrutura geral



Classes de dispositivos genéricos

Dispositivos orientados a caracteres

- Transferências de dados byte a byte, sequencial
- Ex.: dispositivos USB, mouse, teclado, modems

Dispositivos orientados a blocos

- Transferências feitas em blocos de bytes
- Blocos possuem endereços definidos
- Ex.: discos, fitas e dispositivos de armazenamento

Classes de dispositivos genéricos

Dispositivos de rede

- Blocos de dados de tamanho variável (mensagens)
- Envios de blocos de forma sequencial
- NDIS - especificação de interface para dispositivo de rede
- Ex.: Interfaces Ethernet, Buetooth

Dispositivos gráficos

- Renderização de texto e gráficos em uma tela
- Usam uma área de RAM compartilhada (*framebuffer*)
- Acesso por bibliotecas específicas (DirectX, DRI)

Driver

Componente de baixo nível do SO:

- Executa geralmente em modo núcleo
- Interage com o hardware do dispositivo
- Um driver para cada tipo de dispositivo

Funcionalidades:

- Funções de entrada/saída
- Funções de gerência
- Funções de tratamento de eventos

Estrutura de um driver

Funções de **entrada e saída**:

- Transferência de dados dispositivo kernel
- Caracteres, blocos, mensagens

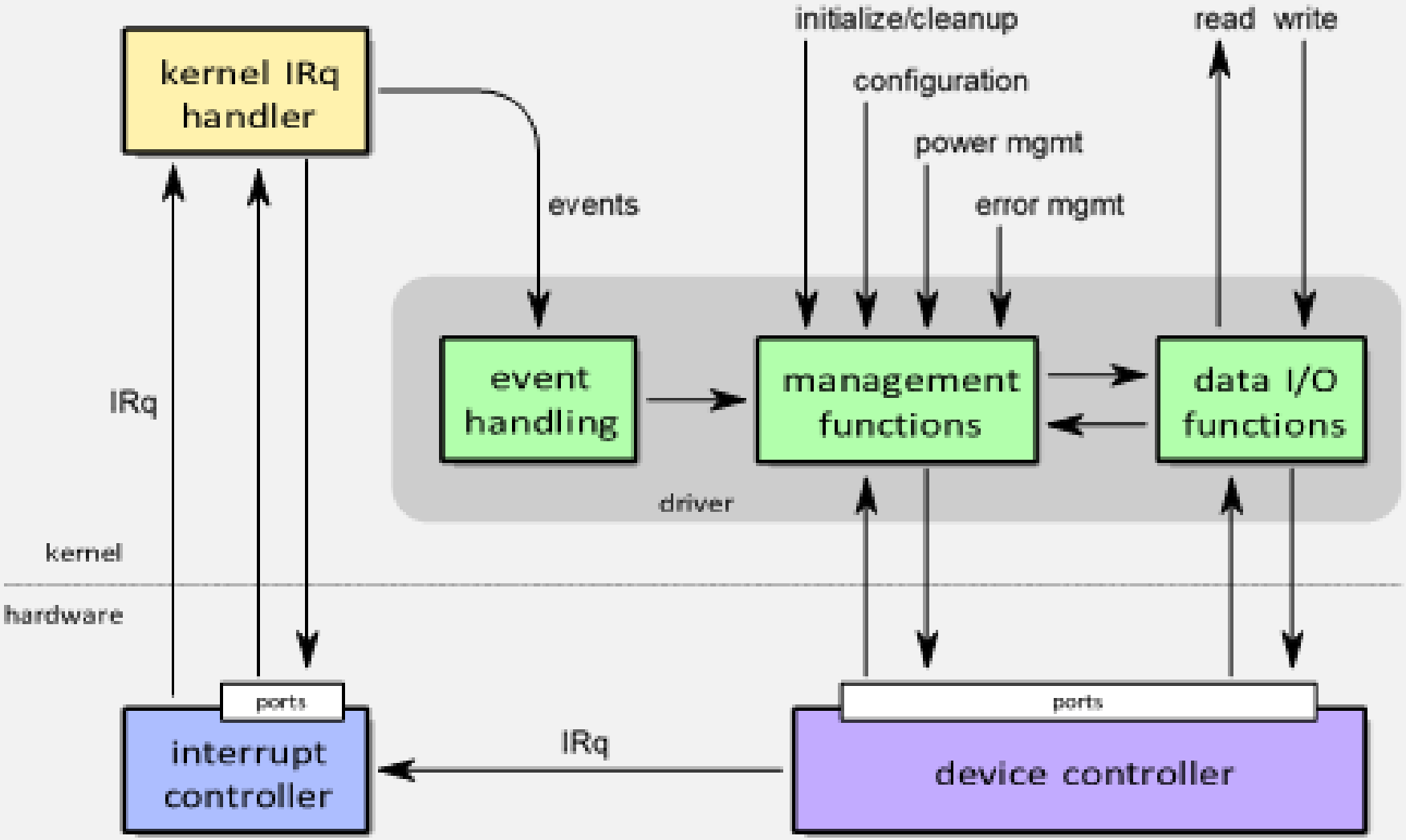
Funções de **gerência**:

- Inicialização e configuração do dispositivo
- Inicialização e configuração do *driver*
- Chamadas de sistema: ioctl(), DeviceControl()

Funções de **tratamento de eventos**:

- Interrupções geradas pelo dispositivo

Estrutura de um driver



Estratégias de interação

Como o kernel interage com os dispositivos?

- Através dos *drivers*!

Os *drivers* implementam **estratégias de interação:**

- Por programa (ou por polling)
- Por eventos (ou por interrupções)
- Por acesso direto à memória (DMA)

Entrada/saída por programa

Estratégia de entrada/saída mais simples.

- Também chamada **varredura** ou **polling**

O *driver* pede a operação e aguarda sua conclusão:

- Espera o dispositivo estar pronto (*status*)
- Escreve dado na porta de saída (*data out*)
- Escreve comando (*control*)
- Espera dispositivo concluir a operação (*status*)

Exemplo: interface paralela simples

I/O port 378H: P0 (porta de dados)

- 8 bits de dados

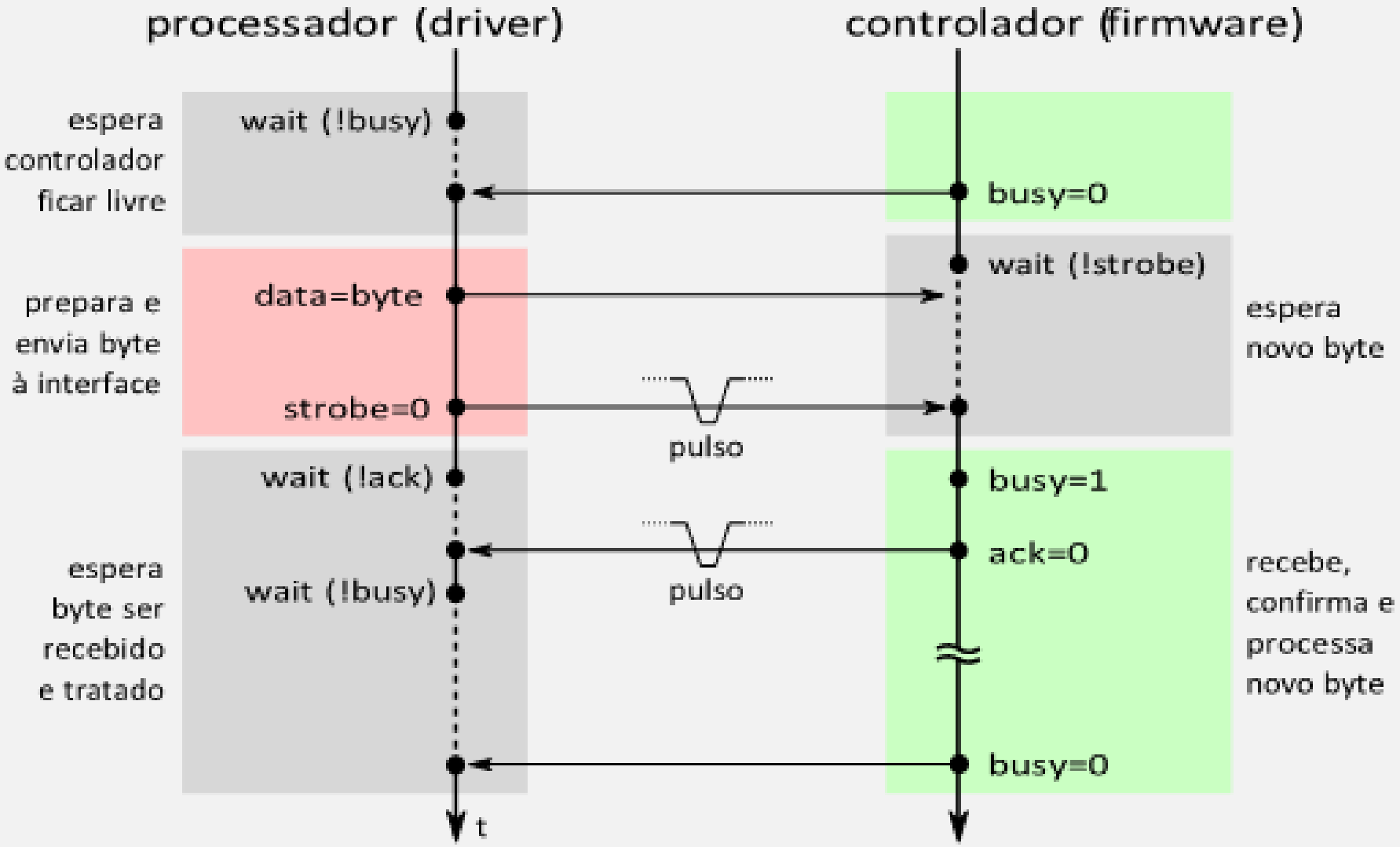
I/O port 379H: P1 (porta de status)

- ack: o dado em P0 foi recebido (acknowledge)
- $\overline{\text{busy}}$: o controlador está ocupado

I/O port 37H: P2 (porta de controle)

- $\overline{\text{strobe}}$: o *driver* diz que há um dado em P_0

Entrada/saída por programa



Entrada/saída por eventos

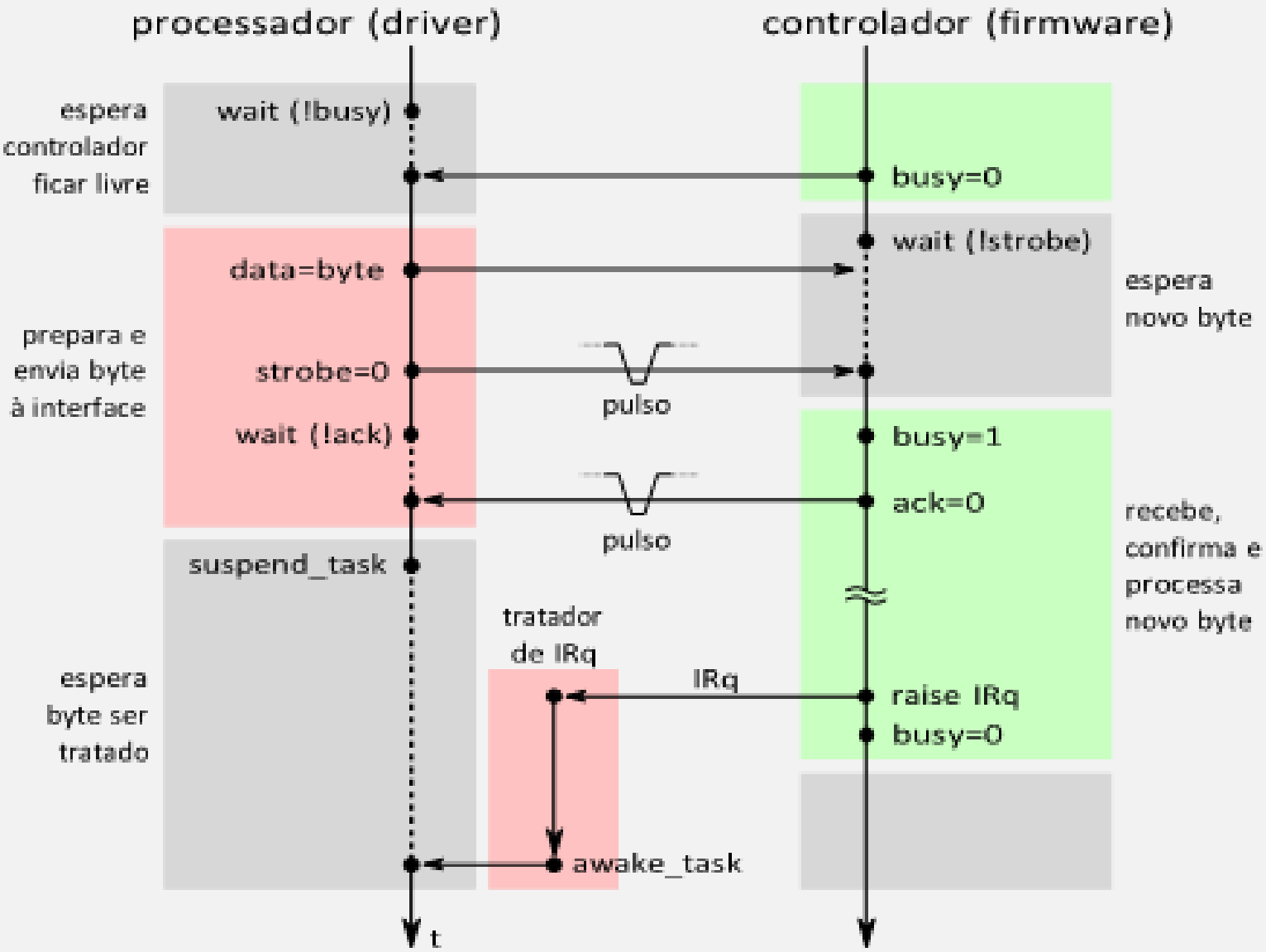
Estratégia básica:

- Requisitar a operação desejada
- Suspender o fluxo de execução (tarefa atual)
- O dispositivo gera uma IRq ao concluir
- O *driver* retoma a operação de E/S

A operação de E/S pelo **driver** é dividida em duas etapas:

- Uma função de E/S inicia a operação
- Uma função de tratamento de evento a continua

Entrada/saída por eventos



Acesso direto à memória

Transferência direta entre dispositivo e RAM

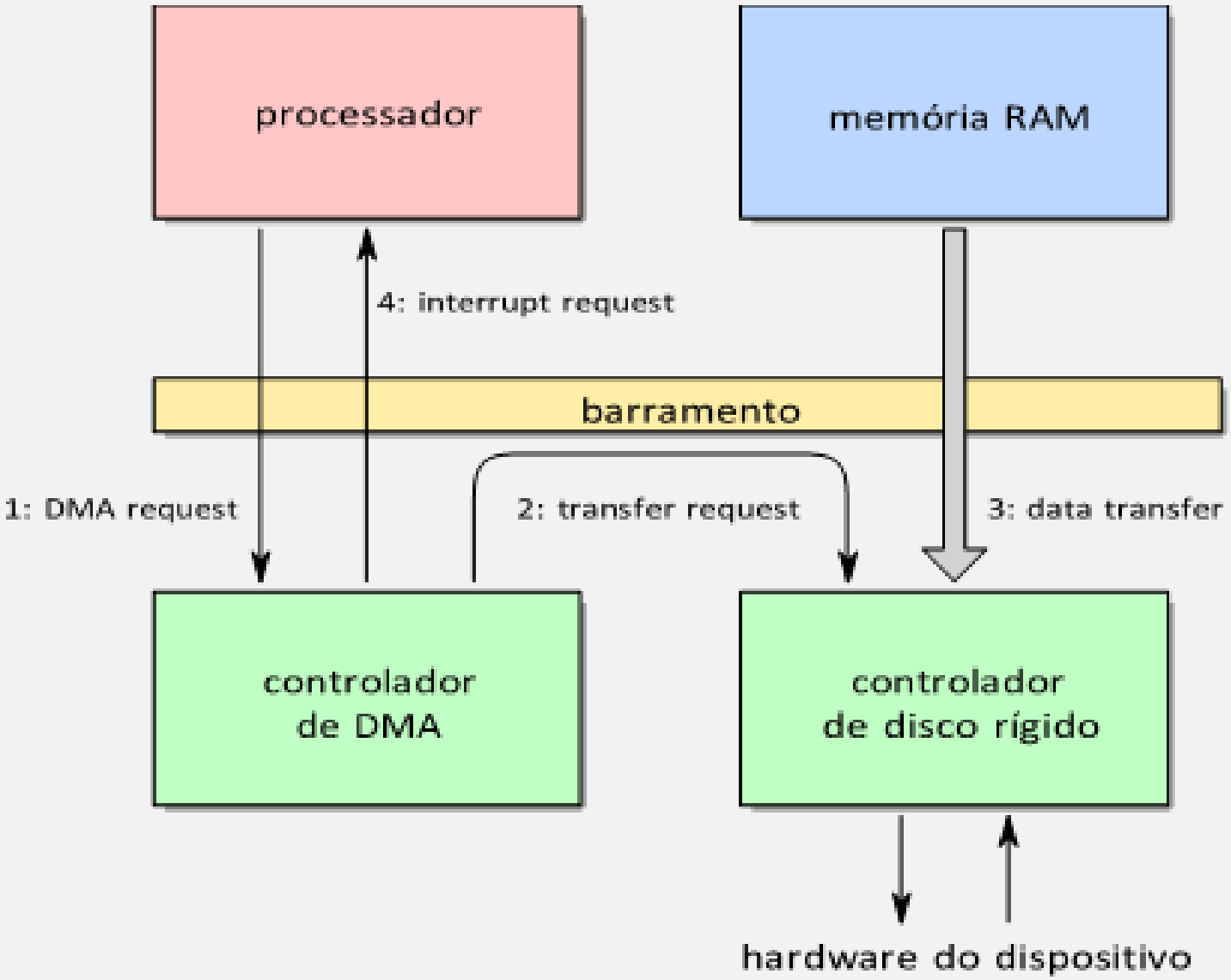
Os dados não precisam passar pela CPU (desempenho)

Muito usado para transferir grandes volumes de dados

Passos:

1. A CPU programa o controlador DMA com os parâmetros (endereços e tamanho da transferência)
2. O controlador de DMA interage com o controlador do para transferir os dados da RAM
3. O controlador do disco recebe os dados da RAM
4. No final, o controlador de DMA interrompe a CPU

Acesso direto à memória



Tratamento de interrupções

Interrupções são tratadas por *handlers* (tratadores)

- Compõem a estrutura dos *drivers*
- Operam usualmente com interrupções inibidas
- Devem ser muito rápidos, portanto simples

A maioria dos SOs trata interrupções em **dois níveis**

Tratamento de interrupções

FLIH - First-Level Interrupt Handler

- Tratador primário (rápido)
- Recebe a IRq e a reconhece junto ao PIC
- Registra dados da Irq em uma fila de eventos
- Outros nomes: Hard, fast, top-half IH

SLIH - Second-Level Interrupt Handler

- Tratador secundário (lento)
- Trata os eventos da fila de eventos
- Pool de threads do kernel escalonadas
- Outros nomes: Soft, slow, bottom-half IH

Tratadores de interrupções FLIH e SLIH