Aula 18 – Dispositivos de Entrada e Saída II

Norton Trevisan Roman
Clodoaldo Aparecido de Moraes Lima

13 de novembro de 2014

E/S – Princípios de Software: Objetivos

- Independência de dispositivo
  - O programador não deve se preocupar com o tipo de dispositivo (CD, HD etc)
  - Cabe ao SO cuidar das idiossincrasias
- Nomenclatura uniforme
  - O nome de um dispositivo deve ser um string ou inteiro, independente do dispositivo
- Tratamento de erro
  - Erros deveriam ser tratados o mais próximo do hardware possível, sem que o usuário tome conhecimento
  - Ex: repetindo a operação

E/S – Princípios de Software

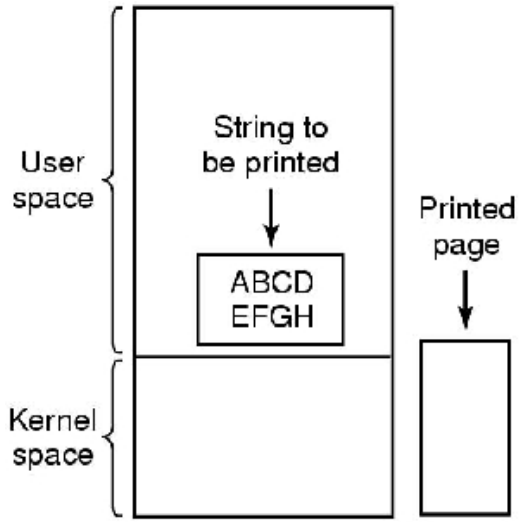
- Modos de execução de E/S:
  - Os módulos de E/S podem operar de 3 modos básicos:
    - E/S programada
    - E/S via Interrupções
    - E/S via Acesso Direto à Memória
  - O que distingue as três formas é a participação da UCP e a utilização das interrupções

E/S – Princípios de Software

- E/S programada
  - Forma mais simples de E/S $\rightarrow$ tudo é feito pela CPU
  - Os dados são trocados entre a CPU e o Módulo de E/S
  - A CPU executa um programa que:
    - Verifica o estado do módulo de E/S, preparando-o para a operação
    - Se necessário, envia o comando que deve ser executado; aguardando o resultado do comando, para então, efetuar a transferência entre o módulo de E/S e algum registrador da CPU

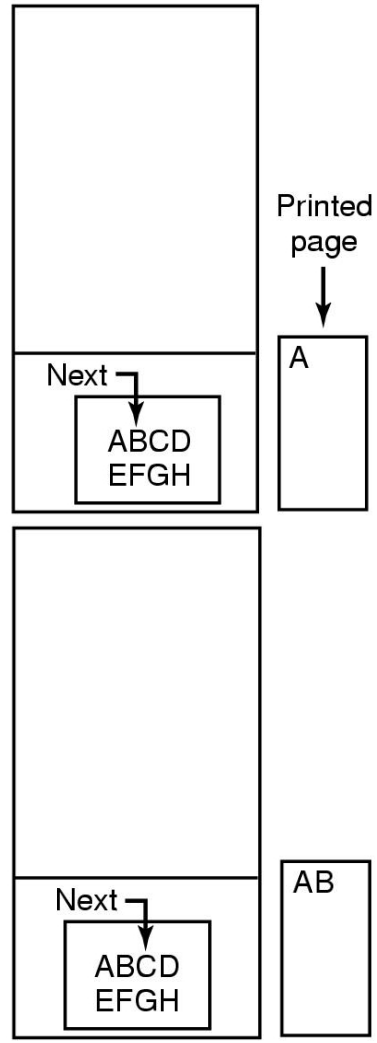
E/S – Princípios de Software

- E/S programada
  - Ex: imprimir "ABCDEFGH" na impressora
    - Monta-se o string em um buffer, no espaço do usuário
    - O processo então detém a impressora (via chamada de sistema – open)
    - Uma vez obtida a impressora, o processo diz ao SO para imprimir (write – chamada ao sistema)
    - O SO copia o string a um arranjo no espaço do kernel
    - Assim que a impressora é liberada, o SO copia o primeiro caractere ao registrador de dados da impressora



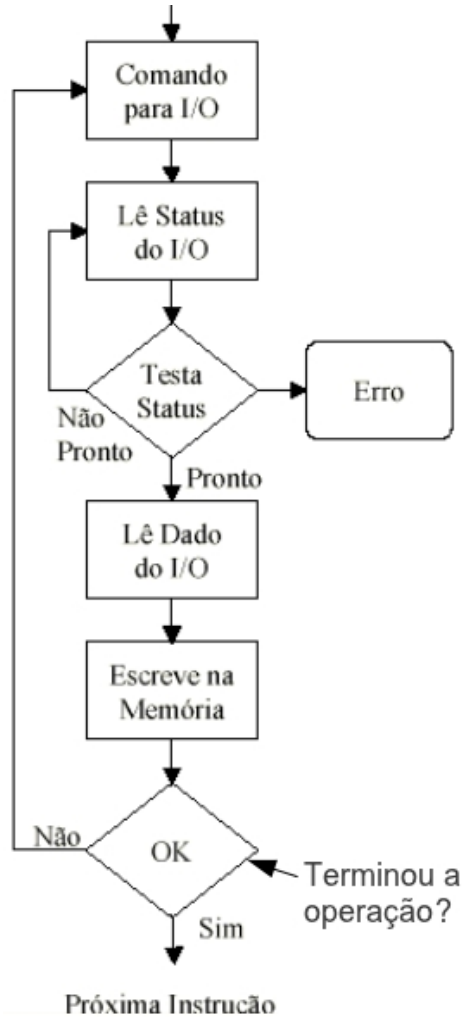
E/S – Princípios de Software

- E/S programada
  - Ex: imprimir "ABCDEFGH" na impressora
    - O primeiro caractere é impresso
    - O sistema marca o próximo caractere, verifica novamente se a impressora está pronta (via registrador de status) para receber outro
    - Se pronta, o novo caractere é enviado
    - Continua até imprimir todo o string
    - Só então o controle volta ao processo



E/S – Princípios de Software

- E/S programada
  - Desvantagem:
    - CPU é ocupada o tempo todo até que a E/S seja feita
    - CPU continuamente verifica se o dispositivo está pronto para aceitar outro caracter → espera ocupada (também chamado de polling)
  - Outro exemplo: Leitura de E/S
    - Escrita é semelhante



Princípios de Software

- E/S via Interrupção
  - Utilizada para superar o problema da espera da CPU por operações nos periféricos
  - Ex: impressora que não armazena caracteres
    - Imprime-os um a um
    - Se imprimir 100/s, cada caractere leva 10ms – tempo suficiente para um chaveamento
    - A impressora, quando está pronta para receber outro caractere, deve gerar interrupção (nesse meio-tempo, a CPU está liberada para fazer outra coisa)

Princípios de Software

- E/S via Interrupção
  - A CPU:
    - Envia um comando para o módulo de E/S e passa a executar outra tarefa
    - Quando a operação for concluída, o módulo de E/S interrompe a CPU
    - A CPU executa a troca de dados, liberando o módulo de E/S e retomando o processamento anterior

Princípios de Software

- E/S via Interrupção
  - Permite que uma unidade ganhe a atenção imediata de outra, de forma que a primeira possa finalizar sua tarefa
  - Geralmente são associados números às interrupções, onde o menor número tem prioridade sobre o maior
  - Ex: Mapeamento das interrupções em um sistema compatível com IBM

Int	Dispositivo	Int	Dispositivo
0	Cronômetro do sistema	9	Porta de comunicação COM3
1	Teclado	10	Porta de comunicação COM2
2	Controlador de interrupção	11	Ponte PCI (*)
4	Porta de comunicação COM1	12	Mouse porta PS/2 (*)
5	Placa de som (*)	13	Coprocessador numérico
6	Controlador de disco flexível	14	Controlador IDE/ESDI
7	Porta de Impressora LPT1	15	Controlador IDE/ESDI
8	CMOS/Relógio do sistema		(*) Opções não padronizadas

Princípios de Software

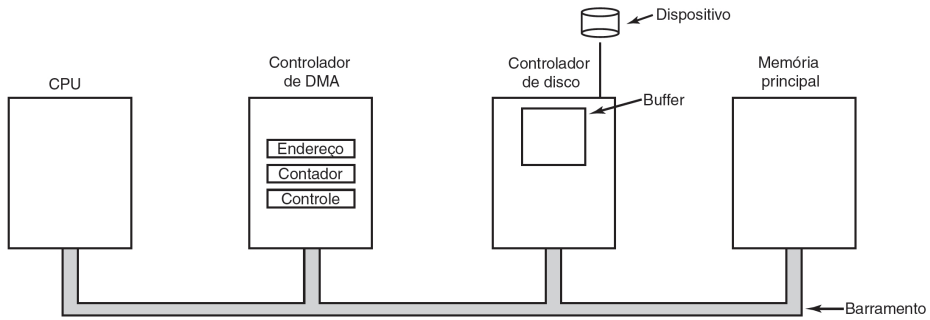
- Inconvenientes dessas técnicas:
  - Limitam a capacidade de transferência entre o módulo de E/S e a Memória Principal
    - A transferência se dá via CPU
    - Trabalham via registradores $\rightarrow$ troca de apenas uma palavra
  - Uso de mais de uma instrução
  - CPU fica ocupada no gerenciamento das interrupções
  - Se a quantidade de dados for grande, o desempenho do sistema será comprometido
    - No caso de interrupção, gera uma a cada caractere

Princípios de Software

- Solução: permitir o acesso direto à memória, tirando o gerenciamento da CPU
  - Direct Memory Access (DMA)
- E/S via Acesso Direto à Memória
  - A CPU não é perturbada
  - Necessita de um controlador de DMA
    - Hardware da placa mãe ou dispositivo
    - Em geral, mais lento que a CPU

Princípios de Software

- E/S via Acesso Direto à Memória
  - Vantagem:
    - Reduz o número de interrupções: de 1 por caractere para uma por buffer, no caso da impressora
  - Deve ter acesso ao barramento independentemente da CPU:
    - Além de registradores que podem ser lidos e escritos pela CPU



E/S via Acesso Direto à Memória

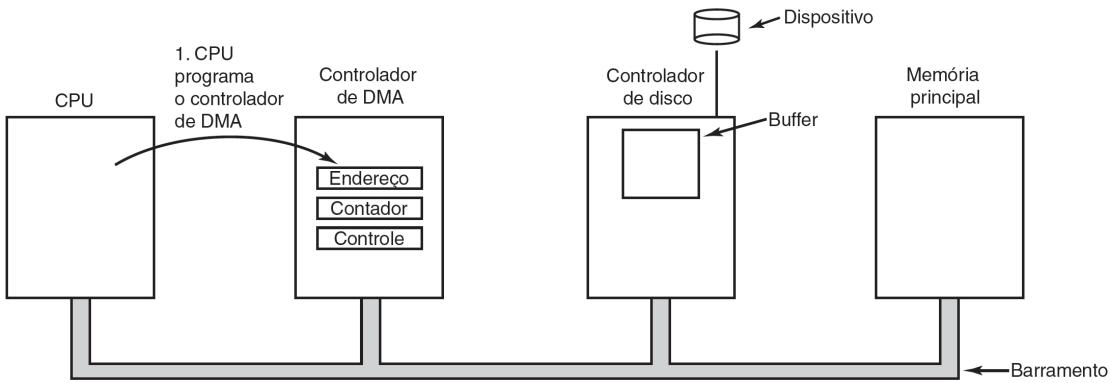
- Registradores:
  - Registrador de endereço de memória (onde armazenar os dados)
  - Contador de bytes
  - Um ou mais registradores de controle, especificando:
    - A porta de E/S em uso
    - Direção da transferência (do dispositivo ou para o dispositivo)
    - Unidade de transferência (byte ou palavra por vez)
    - Número de bytes a serem transferidos em uma única operação (em um surto)

E/S via Acesso Direto à Memória

- E/S sem DMA – disco:
  - A controladora do disco lê um bloco (um ou mais setores) do disco bit a bit, em série
    - Até que o bloco inteiro esteja no buffer interno da controladora
  - Ela calcula o checksum para ver se houve erros de leitura
  - Gera uma interrupção – Quando o SO começa a rodar, pode ler o bloco do buffer da controladora
    - Lê um byte ou palavra por vez, em um laço
    - Cada byte é lido de um registrador na controladora de dispositivo, e armazenado na memória principal

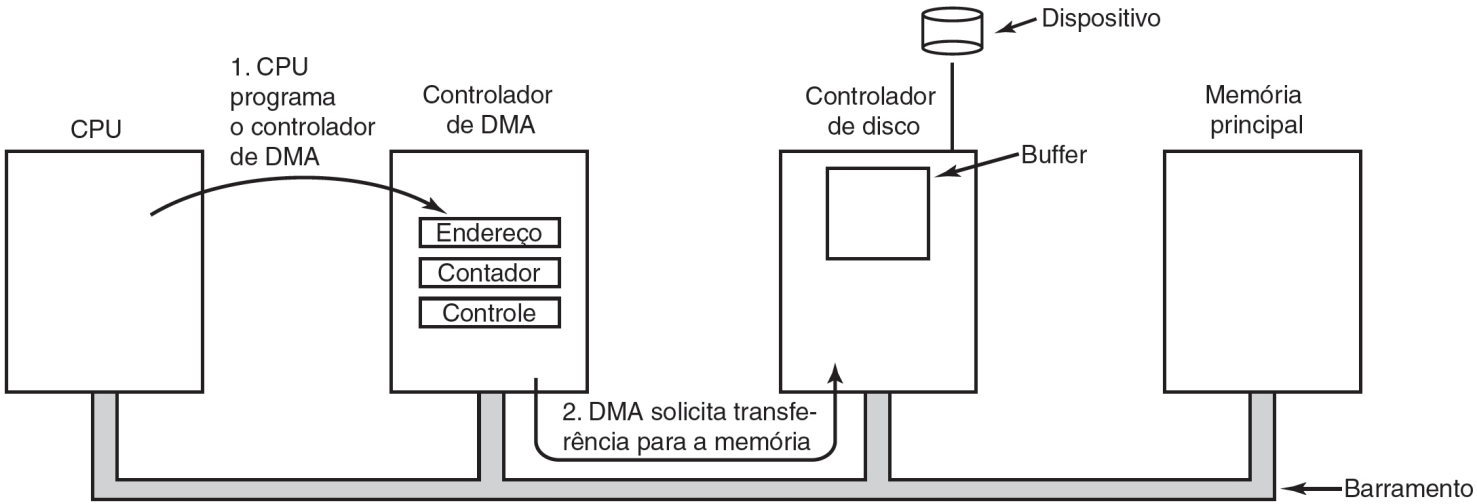
E/S via Acesso Direto à Memória

- E/S com DMA – disco:
  - A CPU programa a controladora de DMA, colocando valores em seus registradores
    - Ele sabe então o que transferir, quanto e para onde
  - CPU continua seu trabalho
  - A controladora do DMA envia um comando à controladora de disco, dizendo-o para ler dados do disco, armazenar em seu buffer interno e verificar o checksum
  - Com dados válidos, no buffer da controladora, DMA pode começar



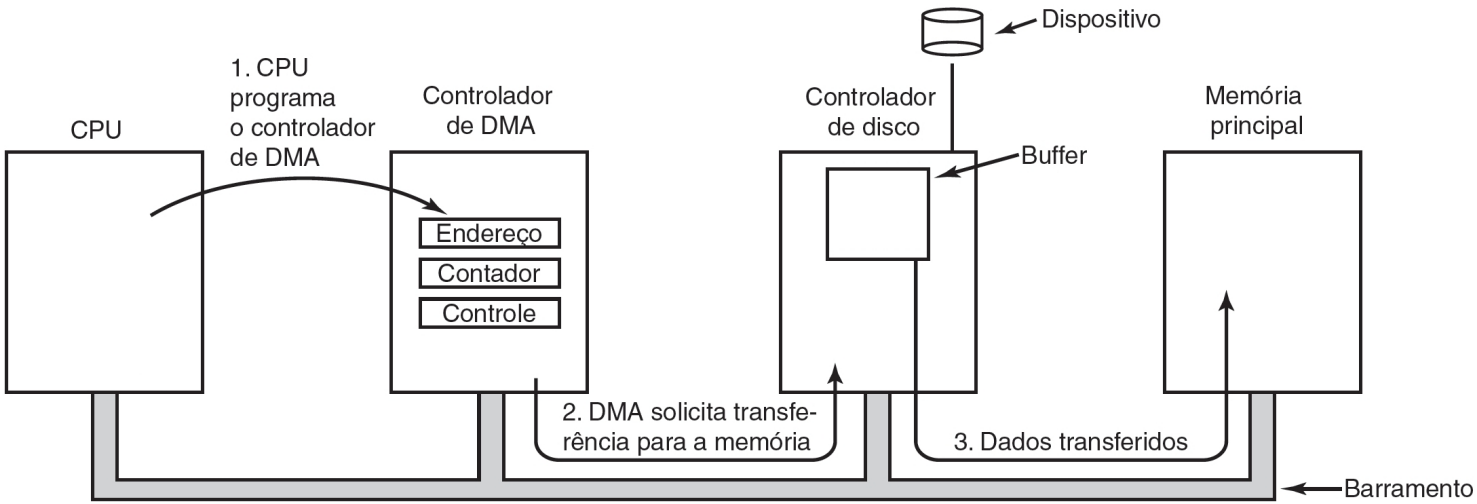
E/S via Acesso Direto à Memória

- E/S com DMA – disco:
  - A controladora de DMA inicia a transferência enviando um pedido de leitura (via barramento) à controladora de disco
    - O endereço de memória onde os dados devem ser armazenados está nas linhas de endereço do barramento



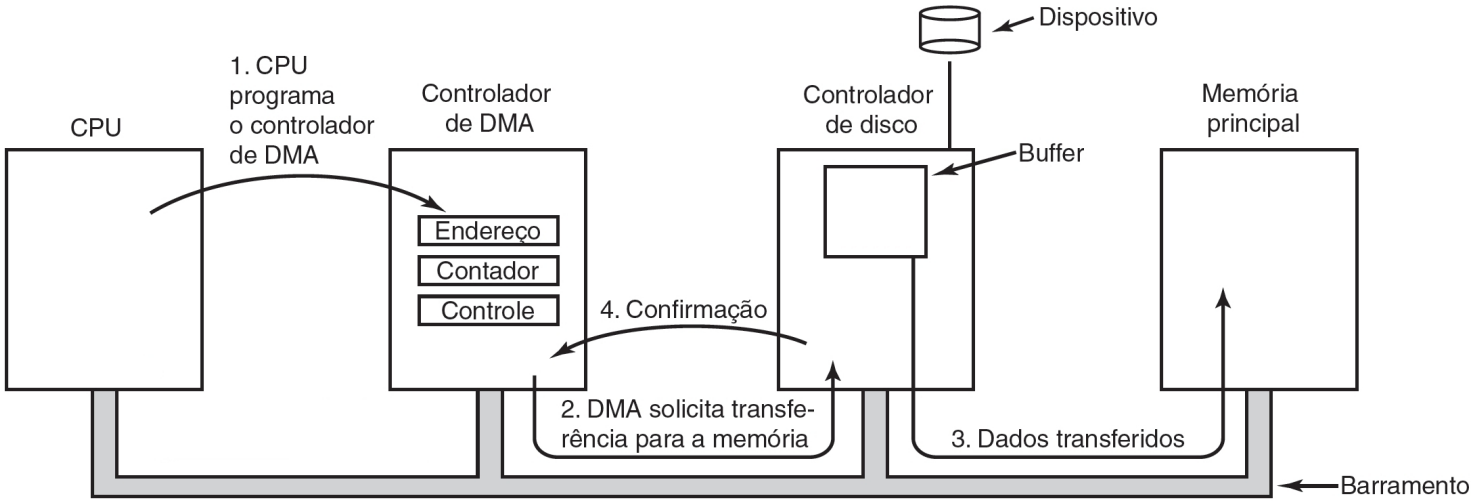
E/S via Acesso Direto à Memória

- E/S com DMA – disco:
  - No próximo ciclo do barramento, automaticamente ocorre a escrita na posição indicada da memória



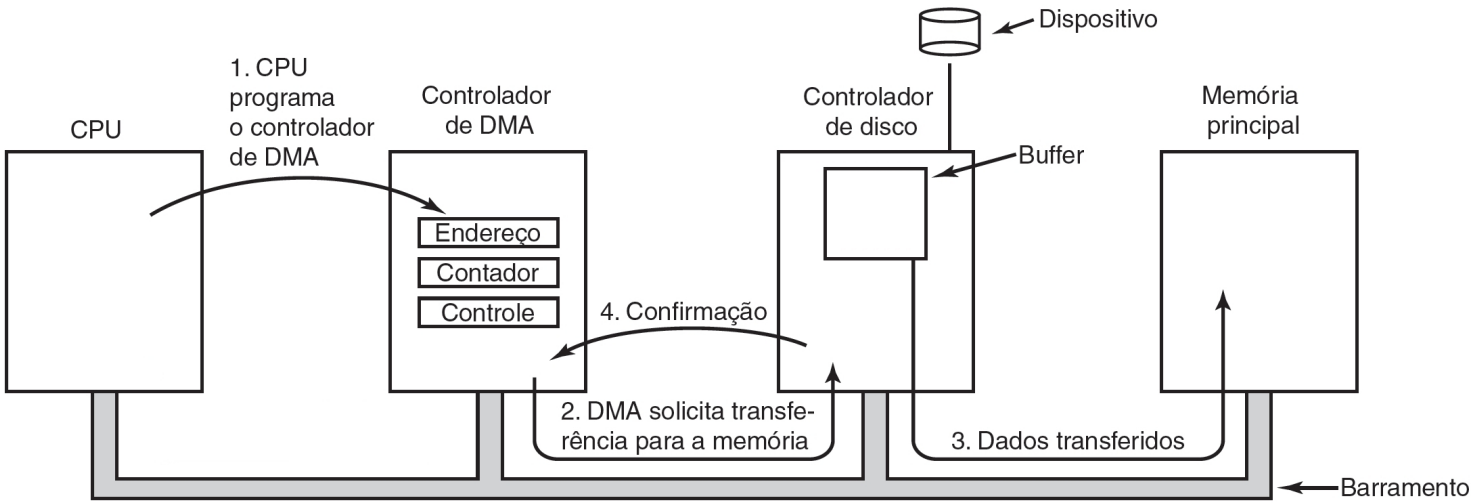
E/S via Acesso Direto à Memória

- E/S com DMA – disco:
  - Quando a escrita é completada, a controladora de disco envia uma confirmação à controladora de DMA, também via barramento



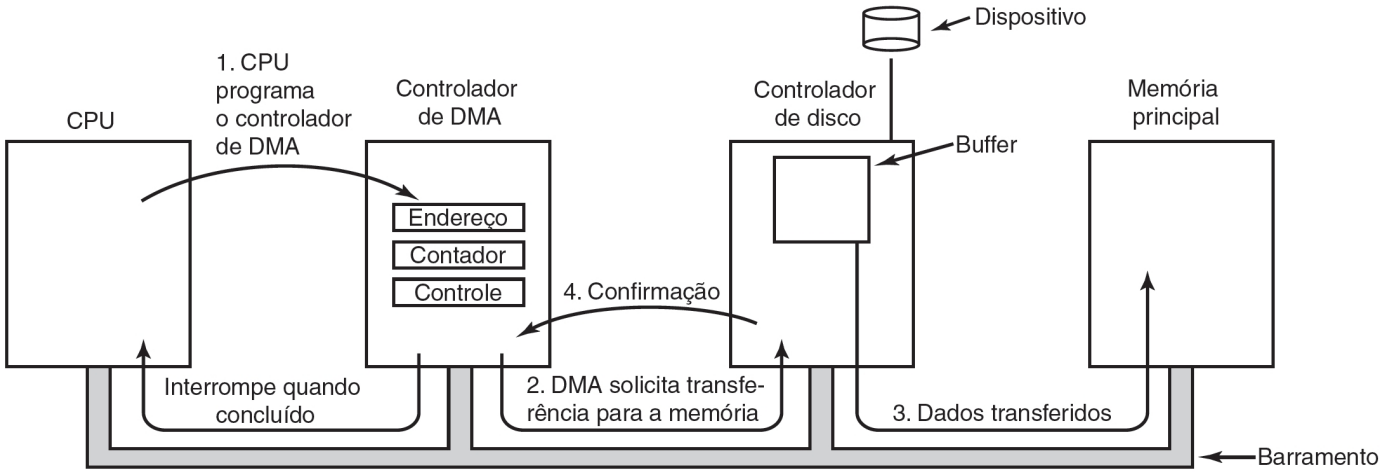
E/S via Acesso Direto à Memória

- E/S com DMA – disco:
  - A controladora de DMA incrementa o endereço de memória a ser usado, diminuindo o contador de bytes
    - Se o contador ainda for maior que 0, repete os passos 2 a 4



E/S via Acesso Direto à Memória

- E/S com DMA – disco:
  - Se o contador for 0, interrompe a CPU
    - A CPU pode executar a rotina de tratamento da interrupção, processando os dados lidos ou produzindo novos dados para serem escritos



  - Quando o SO rodar, não tem que copiar o bloco para a memória – ele já está lá

E/S via Acesso Direto à Memória

- Transferência múltiplas
  - Controladoras de DMA mais sofisticadas podem lidar com vários periféricos diferentes
    - Cada um utilizando um canal de DMA (DMA channel)
    - Devem conter múltiplos conjuntos de registradores, um para cada canal
  - A CPU carrega cada conjunto de registradores com os parâmetros relevantes para a transferência
    - Cada transferência usa uma controladora de dispositivo diferente
  - Após cada transferência de palavra, a controladora de DMA decide quem atender em seguida
    - Escalonamento (round robin, prioridade etc)

E/S via Acesso Direto à Memória

- Roubo de ciclos
  - Durante o acesso à memória, o DMA acessa o barramento
  - Se a CPU quiser usar, terá que esperar
    - Há o roubo ocasional de um ciclo do barramento, atrasando um pouco a CPU
- Há alternativas, mas também com seus problemas:
  - Burst Mode
  - Fly-by mode

E/S via Acesso Direto à Memória

- Modos da controladora – Bust Mode:
  - Em vez de requisitar o barramento a cada palavra, a controladora de DMA:
    - Diz ao dispositivo para segurar o barramento
    - Envia uma série de transferências (um bloco)
    - Libera o barramento
  - Embora mais eficiente, pode bloquear o acesso da CPU ao barramento por um tempo substancial
- Modos da controladora – Fly-by mode
  - Visto até agora – a controladora de DMA diz à controladora de dispositivo para transferir os dados diretamente à memória

E/S via Acesso Direto à Memória

- Alternativamente
  - A controladora do dispositivo envia a palavra à controladora de DMA (usando o barramento)
  - Este então (novamente via barramento), envia à memória ou outro dispositivo
  - Vantagem:
    - Pode executar cópias de dispositivo a dispositivo, sem passar pela memória
  - Problema:
    - Ciclo extra de barramento a cada transferência

E/S via Acesso Direto à Memória

- Desvantagem:
  - A CPU é mais rápida que a controladora de DMA, e pode fazer o trabalho mais rapidamente (se estiver ociosa)
  - Também resulta em arquitetura mais barata
- Vantagem:
  - DMA executa E/S programada $\rightarrow$ controladora de DMA faz todo o trabalho ao invés da CPU
    - Redução do número de interrupções (de uma por byte ou palavra, para uma por buffer)
    - Libera a CPU para executar outras operações

E/S – Princípios de Software

- Camadas de Software de E/S
  - Organizar o software como uma série de camadas facilita a independência dos dispositivos
  - Camadas mais baixas apresentam detalhes de hardware:
    - Drivers e tratadores de interrupção
  - Camadas mais altas fornecem interface para o usuário:
    - Aplicações de Usuário
    - Chamadas de Sistemas
    - Software de E/S Independente do Dispositivo

Drivers

- Cada controladora de dispositivo possui registradores, usados para comunicação com a CPU
- O número de registradores, e sua função, varia conforme o hardware
  - Cada dispositivo de E/S precisa de algum código específico para controlá-lo $\rightarrow$ o driver de dispositivo
    - Contém todo o código dependente do dispositivo
    - É quem efetivamente se comunica com a controladora (inclusive DMA), emitindo comandos e aceitando respostas
  - Dispositivos diferentes possuem drivers diferentes
    - Classes de dispositivos podem ter o mesmo driver

Drivers

- São geralmente escritos pelo fabricante do dispositivo
  - SOs diferentes precisam de drivers diferentes
- Em geral, fazem parte do kernel do SO
  - Permitindo acesso aos registradores da controladora de dispositivo
    - Controlam o funcionamento dos dispositivos por meio de sequência de comandos escritos/lidos nos/dos registradores da controladora
  - Problema;
    - Drivers defeituosos podem causar problemas no kernel do SO

Drivers

- Modos de colocar o driver dentro do kernel:
  - Religar o núcleo com o novo driver, reiniciando o sistema
    - Alguns sistemas UNIX
  - Adicionar uma entrada a um arquivo do SO, informando que ele precisa do driver, reiniciando o sistema
    - Quando inicializa, o SO busca os drivers e os carrega
    - Ex: Windows
  - Fazer com que o SO aceite novos drivers enquanto estiver em execução, instalando-os sem a necessidade de reinicializar
    - Ex: Linux

Drivers

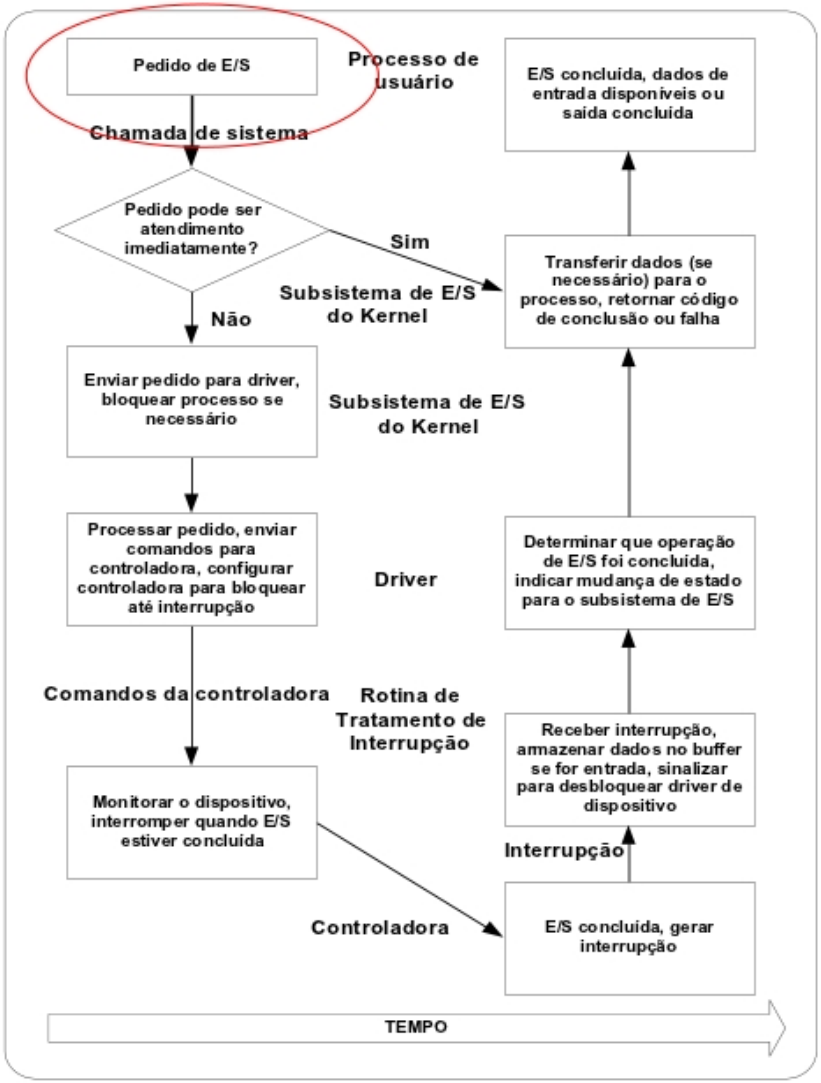
- Funções do SO:
  - Definir um modelo do que um driver deve fazer e como deve interagir com o resto do SO
  - Em geral, define uma interface padrão para:
    - Drivers de dispositivos de bloco
    - Drivers de dispositivos de caracteres
    - Cada interface contém os procedimentos que o resto do SO pode chamar para usar o driver (Ex: ler um bloco, escrever um caractere)
  - Carregar os drivers dinamicamente, durante a execução

Drivers

- Funções do Driver:
  - Aceitar pedidos abstratos de leitura/escrita do software independente de dispositivos e cuidar que sejam executadas
  - Inicializar o dispositivo, se necessário
  - Gerenciar as necessidades energéticas do dispositivo
  - Criar um log de eventos

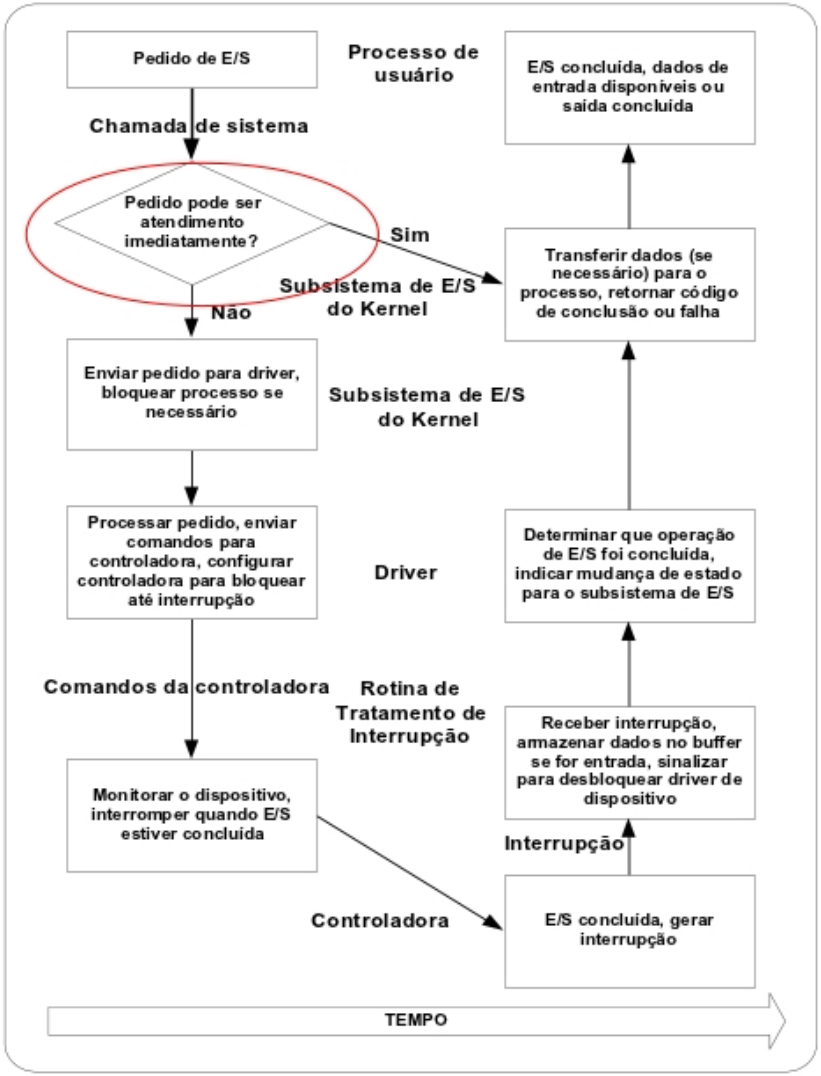
Drivers

- Funcionamento:
  - Um processo emite uma chamada de sistema bloqueante (ex: read) para um arquivo que já está aberto (open)



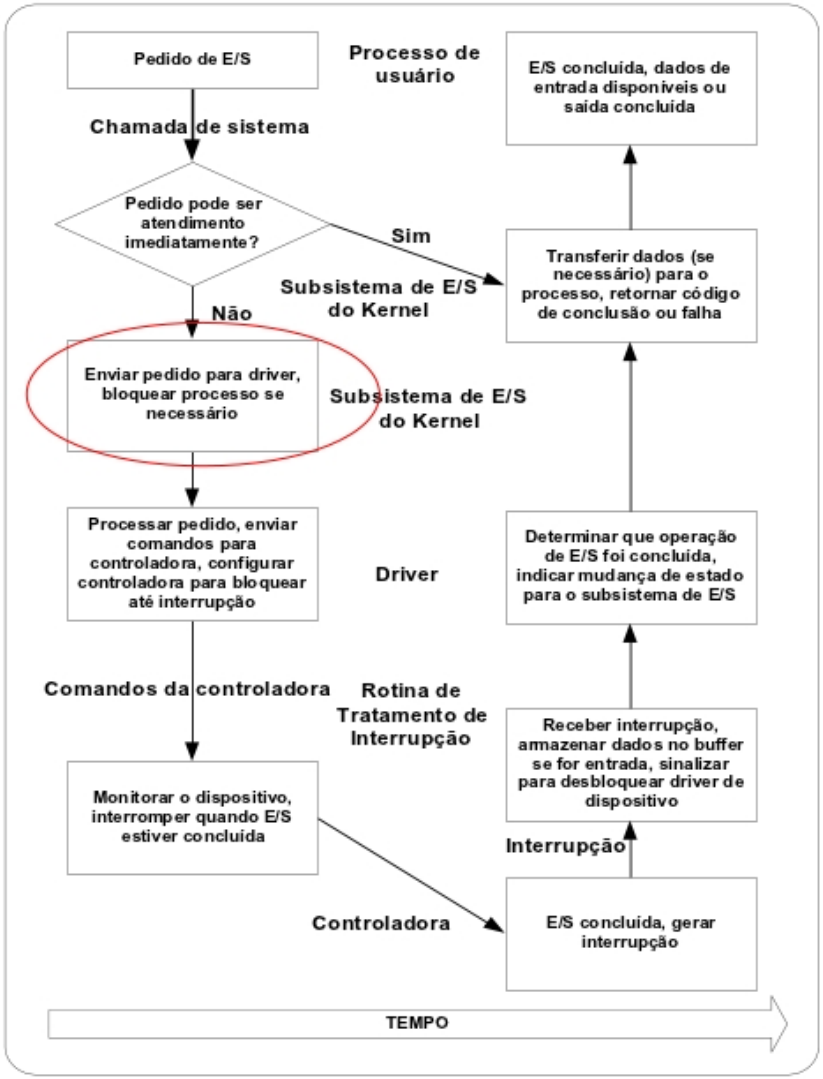
Drivers

- Funcionamento:
  - O código da chamada de sistema verifica os parâmetros. Se os parâmetros estiverem corretos e o arquivo já estiver no buffer (cache), os dados retornam ao processo e a E/S é concluída



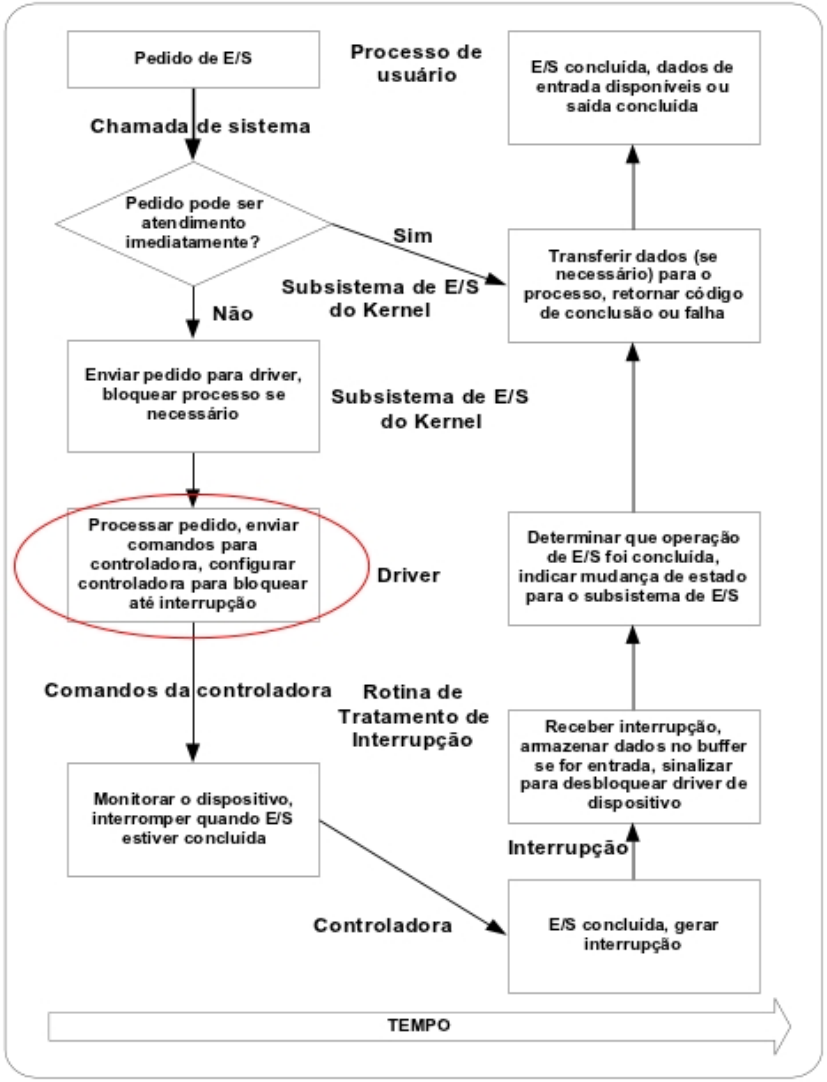
Drivers

- Funcionamento:
  - Se os parâmetros estiverem corretos, mas o arquivo não estiver no buffer, a E/S precisa ser realizada
    - E/S é escalonada
    - Subsistema envia pedido para o driver
  - Se os parâmetros estiverem incorretos, um erro é retornado



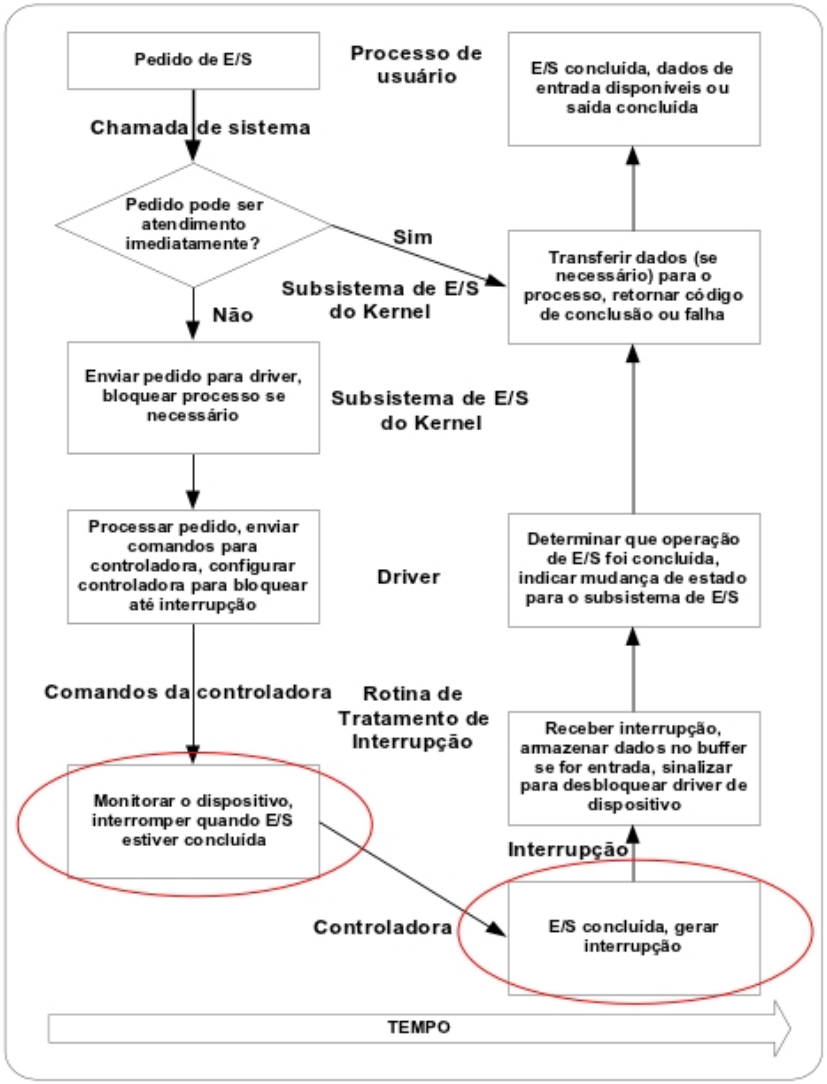
Drivers

- Funcionamento:
  - O Driver aloca espaço de buffer, escalona E/S e envia comando para a controladora do dispositivo escrevendo nos seus registradores de controle
    - Driver pode usar DMA



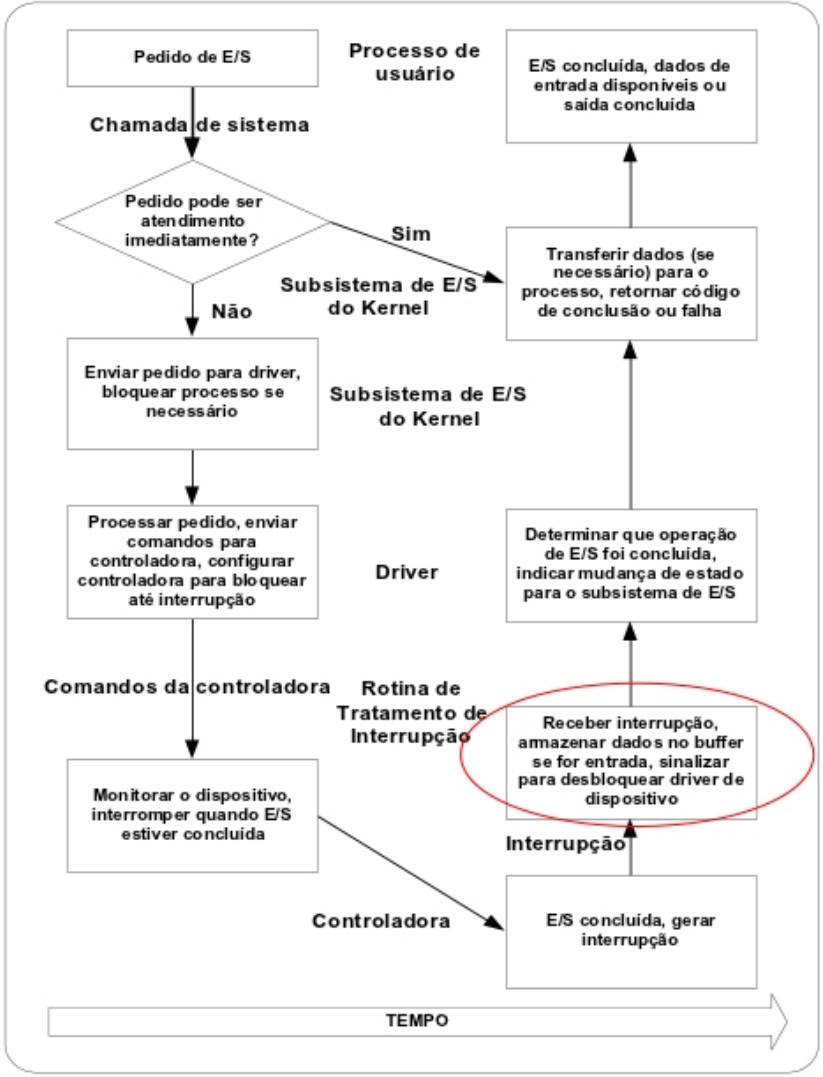
Drivers

- Funcionamento:
  - A controladora do dispositivo opera o hardware, ou seja, o dispositivo propriamente dito
  - Após a conclusão da E/S, uma interrupção é gerada



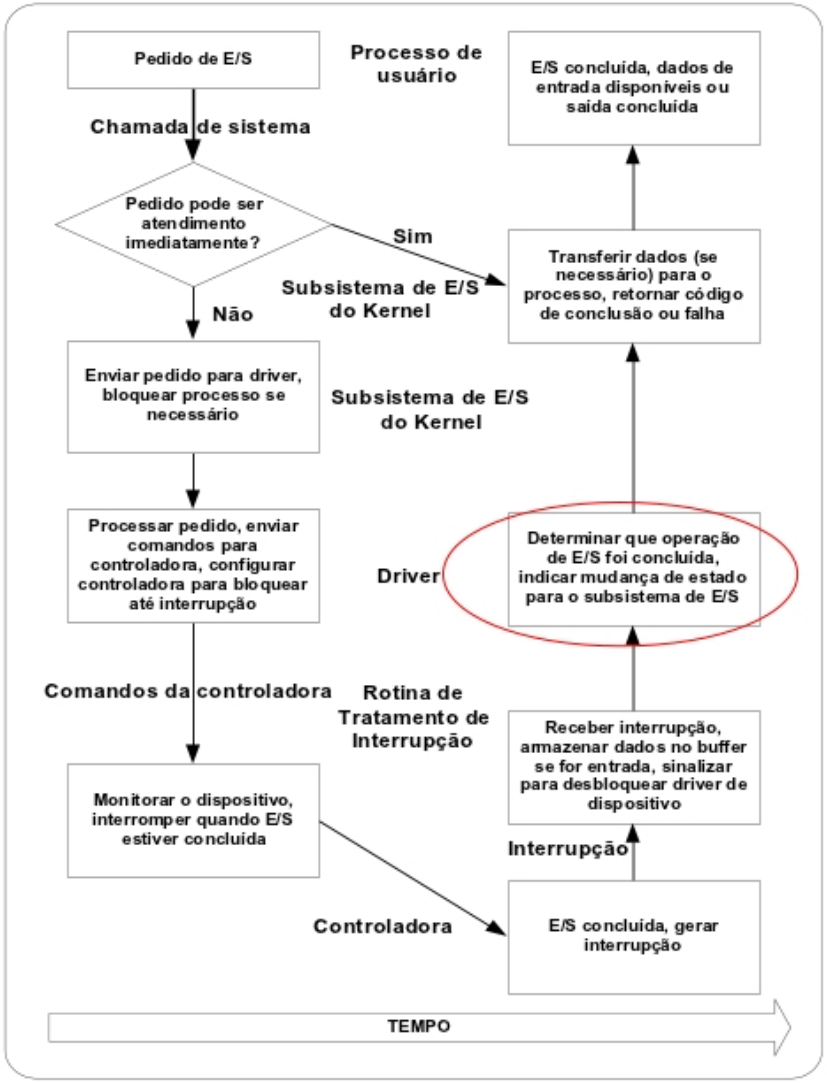
Drivers

- Funcionamento:
  - A rotina de tratamento de interrupções apropriada recebe a interrupção via vetor de interrupção, armazena os dados, sinaliza o driver e retorna da interrupção



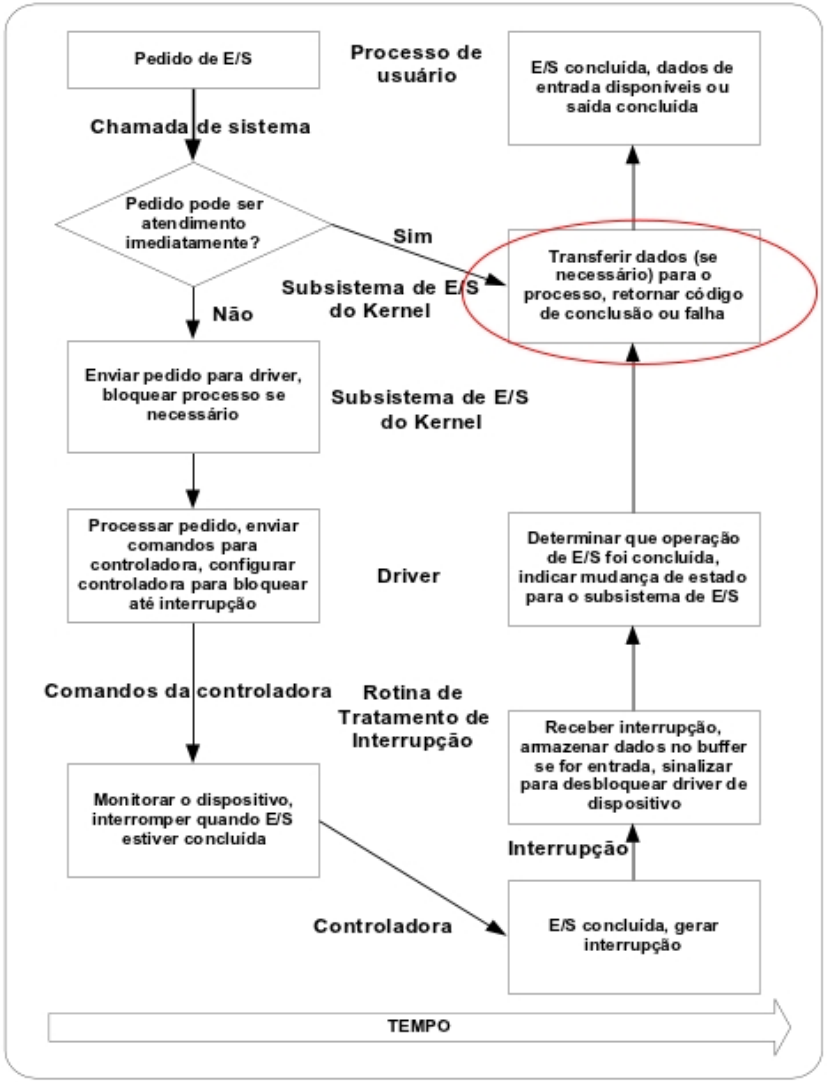
Drivers

- Funcionamento:
  - O Driver recebe o sinal, determina qual pedido de E/S foi concluído, determina o status (se erro ou não) e sinaliza que o pedido está concluído
  - Se outra requisição estiver pendente, poderá agora ser tratada. Senão, o driver bloqueia a si próprio, à espera de nova requisição



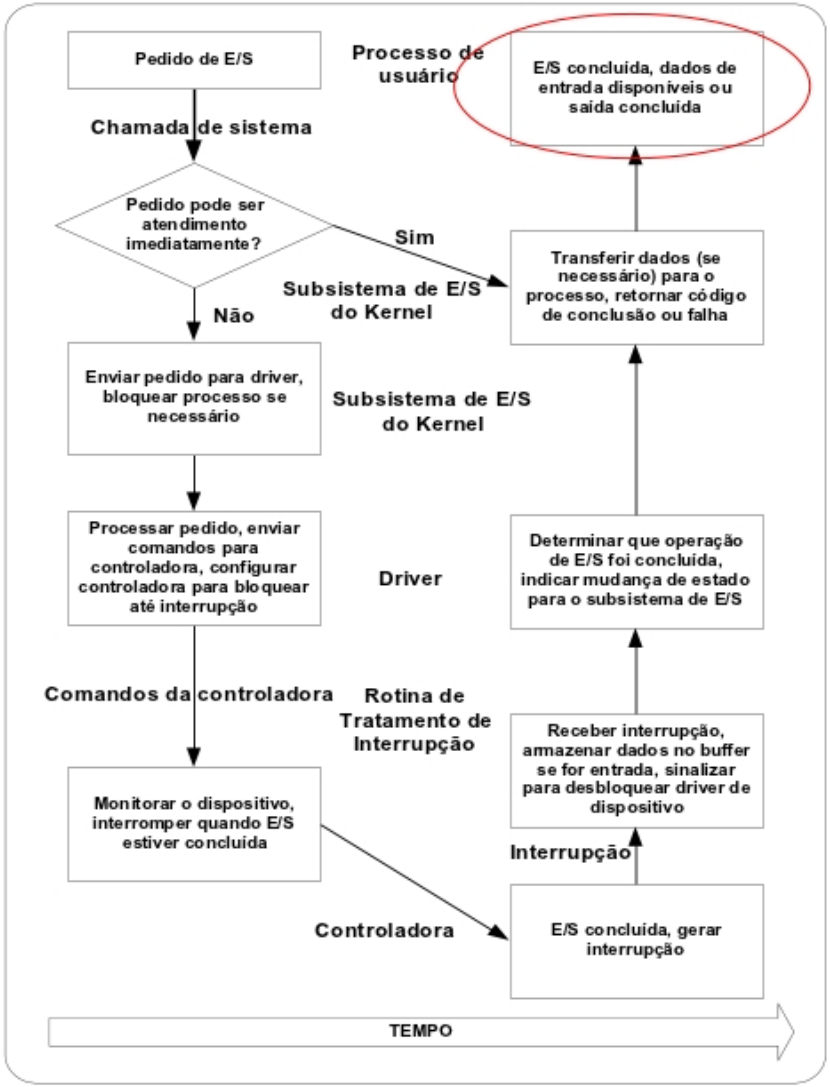
Drivers

- Funcionamento:
  - O Kernel transfere dados ou códigos de retorno para o espaço de endereçamento do processo que requisitou a E/S e move o processo da fila de bloqueados para a fila de prontos



Drivers

- Funcionamento:
  - Quando o escalonador escalona o processo para a CPU, ele retoma a execução na conclusão da chamada ao sistema



Drivers – Reentrância

- Suponha que o dispositivo de E/S completa a tarefa enquanto um driver está executando
  - Interromperá o driver
- A interrupção coloca o mesmo driver em execução
  - Drivers devem ser reentrantes – devem supor que podem ser chamados uma segunda vez, antes que a primeira tenha sido concluída.

Software Independente de Dispositivo

- Algumas partes do software de E/S são específicas e outras independentes de dispositivo
- Sua principal função é executar comandos de E/S comuns a todos os dispositivos
- Outras Funções:
  - Atribuir um nome lógico a partir do qual o dispositivo é identificado. Ex.: UNIX $\rightarrow$ (/dev)
  - Prover buffering:
    - Ajuste entre a velocidade e a quantidade de dados transferidos;

Software Independente de Dispositivo

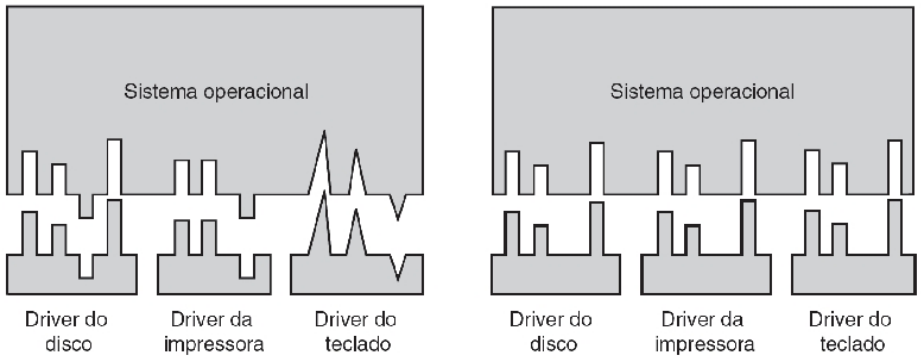
- Outras Funções
  - Fazer o escalonamento de E/S
  - Cache de dados
    - Armazenar na memória um conjunto de dados frequentemente acessados
  - Gerenciar alocação, uso e liberação dos dispositivos
    - Acessos concorrentes tanto a recursos compartilháveis (que podem ser utilizados por vários usuários ao mesmo tempo – disco etc) quanto a dedicados (que podem ser utilizados por apenas um usuário de cada vez – impressora etc)
    - Ex: via chamadas open a arquivos especiais, associados aos dispositivos – se este não estiver disponível, o open falha

Software Independente de Dispositivo

- Outras Funções
  - Reportar erros e proteger os dispositivos contra acessos indevidos:
    - Erros de programação. Ex.: tentar efetuar leitura de um dispositivo de saída (impressora, vídeo)
    - Erros de E/S. Ex.: tentar imprimir em uma impressora desligada ou sem papel
    - Erros de memória. Ex.: Escrita em endereços inválidos
    - Ações: retornar código de erro ao processo chamador, tentar solucionar no próprio dispositivo etc
  - Definir tamanhos de blocos independentes do dispositivo, fornecendo um tamanho uniforme para as camadas superiores

Software Independente de Dispositivo

- Outras Funções
  - Fornecer interface uniforme ao software do usuário:
    - Evita que a cada novo dispositivo criado, o SO tenha que ser modificado.
    - Uniformiza a interface entre o SO e seus drivers de dispositivos – cada dispositivo fornece driver com funções diferentes $\times$ há padrão nas funções fornecidas
    - Nesse último caso, para cada classe de dispositivos, o SO define um conjunto de funções que o driver deve fornecer



Sem uma interface-padrão para o driver. Com uma interface-padrão para o driver.