# *Curso de Engenharia de Computação*
# *Sistemas Operacionais*

## Entrada e Saída

Slides da disciplina Sistemas Operacionais
Curso de Engenharia de Computação
Instituto Mauá de Tecnologia – Escola de Engenharia Mauá
Prof. Marco Antonio Furlan de Souza

# *Dispositivos de E/S*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Tipos de dispositivos de E/S**
  - **Dispositivo de bloco**: armazena informações em **blocos** de **tamanho fixo**, cada um com seu próprio endereço. Os **tamanhos** de **bloco** comuns **variam** de **512** a **65536 bytes**. Todas as **transferências** estão em **unidades** de **um** ou **mais blocos** inteiros (consecutivos). A **propriedade essencial** de um dispositivo de bloco é que é **possível ler** ou **gravar** cada **bloco independentemente** de todos os outros. **Discos rígidos**, **discos Blu-ray** e **pen drives** são **dispositivos** de **bloco** comuns.
  - **Dispositivo de caractere**: **entrega** ou **aceita** um **fluxo** de **caracteres**, **sem considerar** nenhuma **estrutura** de bloco. **Não** é **endereçável** e **não possui nenhuma operação** de **busca**. **Impressoras**, **interfaces** de **rede**, **mouses** (para apontar) e a **maioria** dos **outros** dispositivos **que não são** do **tipo disco** podem ser vistos como **dispositivos** de **caracteres**

# *Dispositivos de E/S*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Tipos de dispositivos de E/S**
  - Alguns dispositivos não se encaixam nesta classificação:
    - Os **relógios** (clock), por exemplo, **não** são **endereçáveis** por **bloco** e **nem geram** ou **aceitam fluxos** de **caracteres**. Tudo o que eles fazem é **causar interrupções** em **intervalos** bem **definidos**;
    - As **telas mapeadas** na **memória** e **telas sensíveis** ao **toque** também não se encaixam bem no modelo.

# *Dispositivos de E/S*

INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

- **Dispositivos típicos**
  - Dispositivos e taxas de transmissão no *bus*

| Device | Data rate |
|---|---|
| Keyboard | 10 bytes/sec |
| Mouse | 100 bytes/sec |
| 56K modem | 7 KB/sec |
| Scanner | 400 KB/sec |
| Digital camcorder | 3.5 MB/sec |
| 802.11g Wireless | 6.75 MB/sec |
| 52x CD-ROM | 7.8 MB/sec |
| Fast Ethernet | 12.5 MB/sec |
| Compact flash card | 40 MB/sec |
| FireWire (IEEE 1394) | 50 MB/sec |
| USB 2.0 | 60 MB/sec |
| SONET OC-12 network | 78 MB/sec |
| SCSI Ultra 2 disk | 80 MB/sec |
| Gigabit Ethernet | 125 MB/sec |
| SATA disk drive | 300 MB/sec |
| Ultrium tape | 320 MB/sec |
| PCI bus | 528 MB/sec |

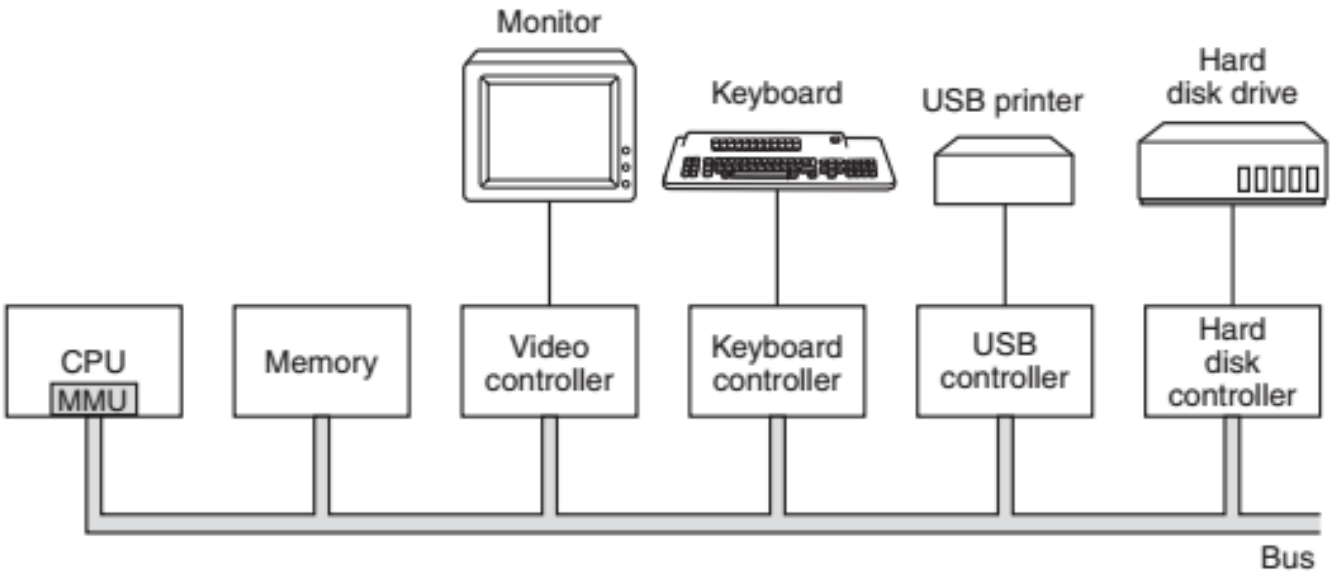
# *Controladores de dispositivos de E/S*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - As **unidades** de **E/S** geralmente **consistem** em um **componente mecânico** e um **componente eletrônico**;
  - É possível **separar** as **duas** porções para fornecer uma **forma** mais **modular**:
    - O **componente eletrônico** é chamado de **controlador de dispositivo** ou **adaptador**;
    - O **componente mecânico** é o **dispositivo** propriamente dito.
  - A **interface** entre o **controlador** e o **dispositivo** é normalmente descrita por um **padrão**: SATA, SCSI, USB, Thunderbolt, FireWire (IEEE 1394) são alguns exemplos de interface.

# *Controladores de dispositivos de E/S*

- **Conceitos**
  - **Controladores × Dispositivos**



# *E/S Mapeada em Memória*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - Cada **controlador** possui alguns **registradores** usados para se **comunicar** com a **CPU**;
  - Ao **escrever** nesses **registradores**, o **sistema operacional** pode **comandar** o **dispositivo** para **fornecer dados**, **aceitar dados**, **ativar** ou **desativar** ou ainda **executar** alguma **ação**;
  - Ao **ler** desses **registros**, o **sistema operacional** pode **aprender** em que **estado** o **dispositivo** está, **se** ele está **preparado** para **aceitar** um novo **comando** e **assim** por diante;
  - Muitos **dispositivos** possuem um **buffer de dados** que o **sistema operacional pode ler** e **gravar** (exemplo: RAM de vídeo).

# *E/S Mapeada em Memória*

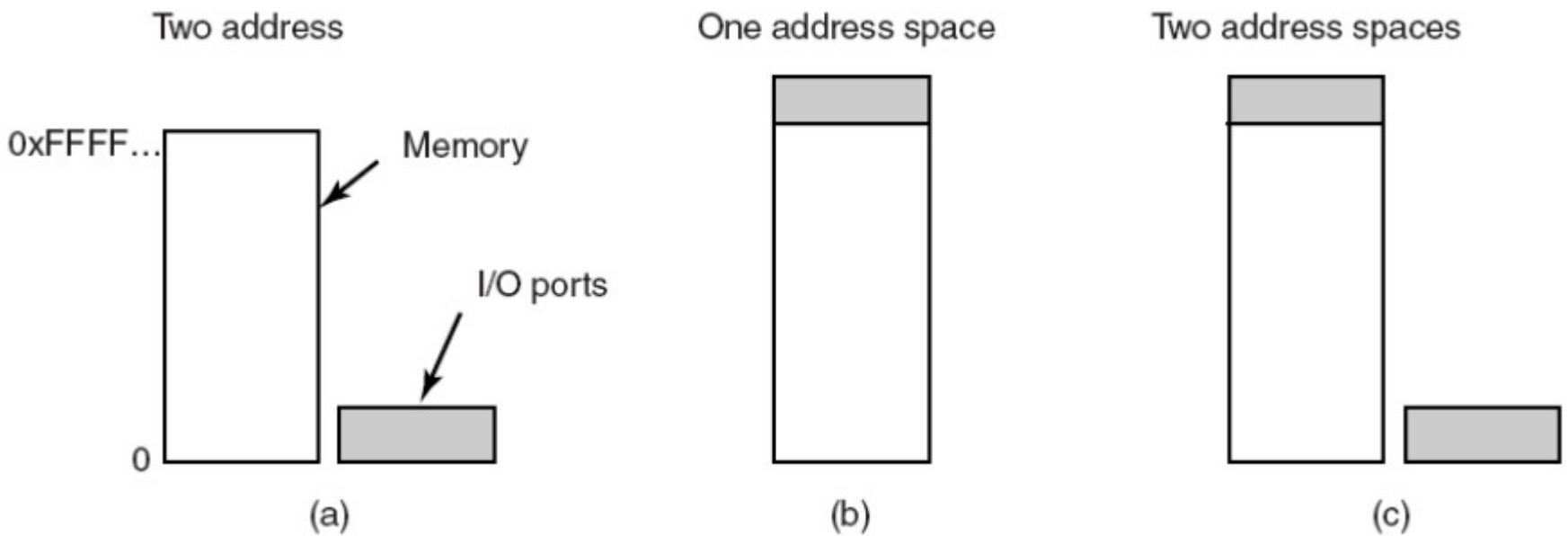
INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - Como a **CPU** se **comunica** com os **registradores** de **controle** e com os **buffers** de **dados** dos **dispositivos**?
    - **Solução 1**: a cada **registrador** de **controle** é **atribuído** um **número de porta** de **E/S** – um **número inteiro** de 8 ou 16 bits – e o conjunto de **todas** as **portas** de **E/S forma** o **espaço de portas de E/S**, que é **protegido** para que os **programas** do **usuário não possam acessá-lo** (somente o sistema operacional pode). Neste esquema, o **espaço** de **endereçamento** de **memória** e de **E/S** são **separados**;
    - **Solução 2**: **mapear** todos os **registradores** de **controle** no **espaço** de **memória**. Isto é chamado de **E/S mapeada em memória**.

# E/S Mapeada em Memória

- **Tipos de mapeamento**

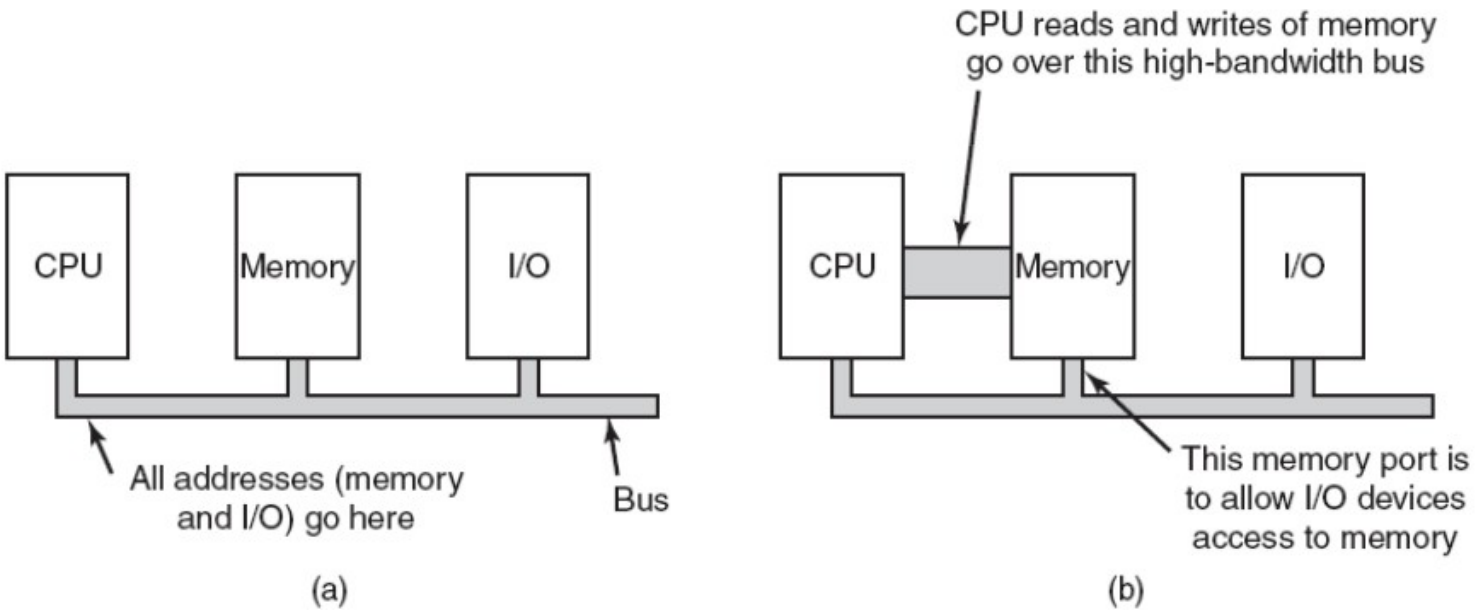
  (a) E/S separada da memória; (b) E/S mapeada em memória; (c) Híbrido



# *E/S Mapeada em Memória*

- **Tipos de *bus* para E/S mapeada em memória**

  (a) *Bus* único para memória e E/S; (b) *Bus* de alta velocidade para CPU/memória e *bus* para dispositivos de I/O para acessar a memória



# *Acesso direto à memória (DMA)*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - **Independentemente** de uma **CPU ter ou não E/S mapeada** na **memória**, ela precisa **endereçar** os **controladores** do **dispositivo** para **trocar dados** com eles;
  - A **CPU** pode **solicitar dados** de um controlador de E/S, **um byte de cada vez**, mas **isso desperdiça** o **tempo** da **CPU**;
  - **Portanto**, um **esquema diferente**, **chamado DMA** (*Direct Memory Access*), é frequentemente **usado**;
  - O sistema operacional pode **usar** apenas **DMA** se o **hardware tiver** um **controlador DMA**, o que a maioria dos sistemas possui (exemplo: integrado a discos);
  - Mais **comumente**, um **único controlador DMA** está **disponível** (por exemplo, na **placa-mãe**) para **regular transferências** para **vários dispositivos**, geralmente simultaneamente.

# *Acesso direto à memória (DMA)*

INSTITUTO MAUÁ DE TECNOLOGIA

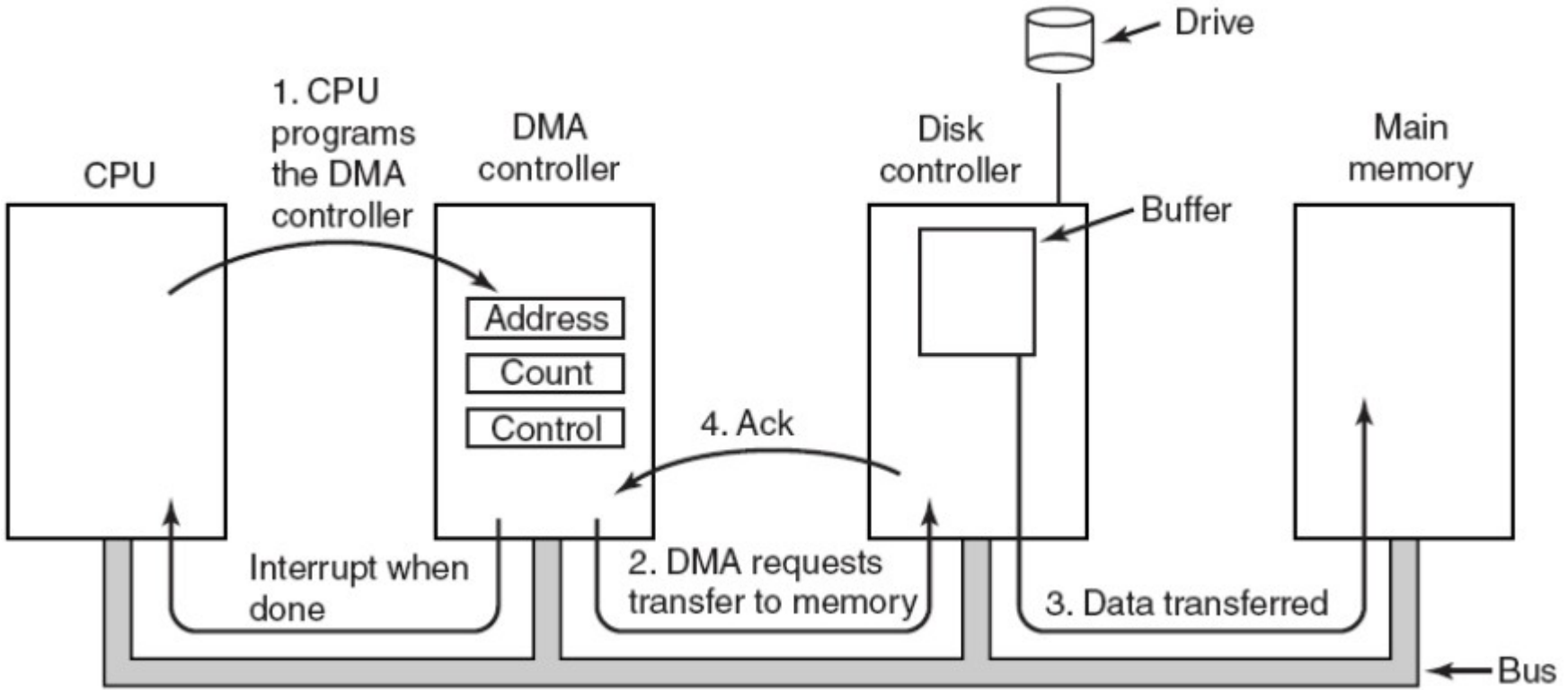
- **Como funciona o DMA?**
  - Por exemplo, quando se lê de um disco:
    1) Primeiro, a **CPU programa** o **controlador DMA**, **configurando** seus **registradores** para **saber o que transferir para onde**. Ela **também emite** um **comando** para o **controlador de disco**, **solicitando** que ele **leia** os **dados** do **disco** em seu **buffer interno** e **verifique** a **soma de verificação** (*checksum*). Quando **dados válidos** estão no **buffer** do **controlador** de **disco**, o **DMA** pode **começar**;
    2) O **controlador DMA** inicia a **transferência emitindo** uma **solicitação** de **leitura** pelo **barramento** para o **controlador** de **disco** – o **controlador** de **disco não sabe se veio** da **CPU** ou de um **controlador DMA**;
    3) A **gravação** na **memória** é outro **ciclo** de **barramento padrão**;
    4) Quando a **gravação** é **concluída**, o **controlador** de **disco envia** um sinal de **confirmação** para o **controlador DMA**, também através do barramento.
    5) No fim, o **DMA interrompe** a **CPU** para **informar** que a **transferência** está **concluída**.

Repete até transferir todos os dados

# *Acesso direto à memória (DMA)*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Operação de transferência com DMA**



# *Interrupções*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - **Quando** um **dispositivo** de **E/S conclui** o **trabalho** que lhe é dado, **causa** uma **interrupção** (assumindo que as interrupções foram ativadas pelo sistema operacional);
  - Isso se **inicia** por um **sinal** na **linha** do ***bus*** que lhe foi atribuída. Esse **sinal** é **detectado** pelo **chip** do **controlador** de **interrupção** na **placa-mãe**, que **decide** o **que fazer**.
  - Se **nenhuma outra interrupção** estiver **pendente**, o **controlador** de interrupção **tratará** a **interrupção imediatamente**. No **entanto**, se **outra** interrupção **estiver** em **andamento** ou outro **dispositivo** tiver **feito** uma **solicitação simultânea** em uma **linha** de **solicitação** de **interrupção** de **prioridade mais alta** no barramento, o **dispositivo** será **ignorado** no **momento**. **Nesse caso**, ele **continua** a **gerar** um **sinal** de **interrupção** no **barramento até** que seja **atendido** pela CPU.

# *Interrupções*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - Para **lidar** com a **interrupção**, o **controlador coloca** um **número** nas **linhas** de **endereço**, **especificando** qual **dispositivo** deseja **atenção** e **emite** um **sinal** para **interromper** a **CPU**;
  - O **sinal** de **interrupção** faz com que a **CPU pare** o **que está fazendo** e **comece** a **fazer outra coisa**;
  - O **número** nas **linhas** de **endereço** é usado como um **índice** em uma **tabela** chamada **vetor** de **interrupção** para **buscar** um **novo contador** de **programa** – esse **contador** de **programa aponta** para o **início** do **procedimento** de **serviço** de interrupção **correspondente**;
  - Normalmente, os **traps** e **interrupções usam** o **mesmo mecanismo** a **partir deste ponto**, geralmente **compartilhando** o **mesmo vetor** de **interrupção**. O **local** do **vetor** de interrupção pode ser **conectado** à **máquina ou** em **qualquer lugar** da **memória**, como um **registro** da **CPU** (carregado pelo sistema operacional) apontando para sua origem.

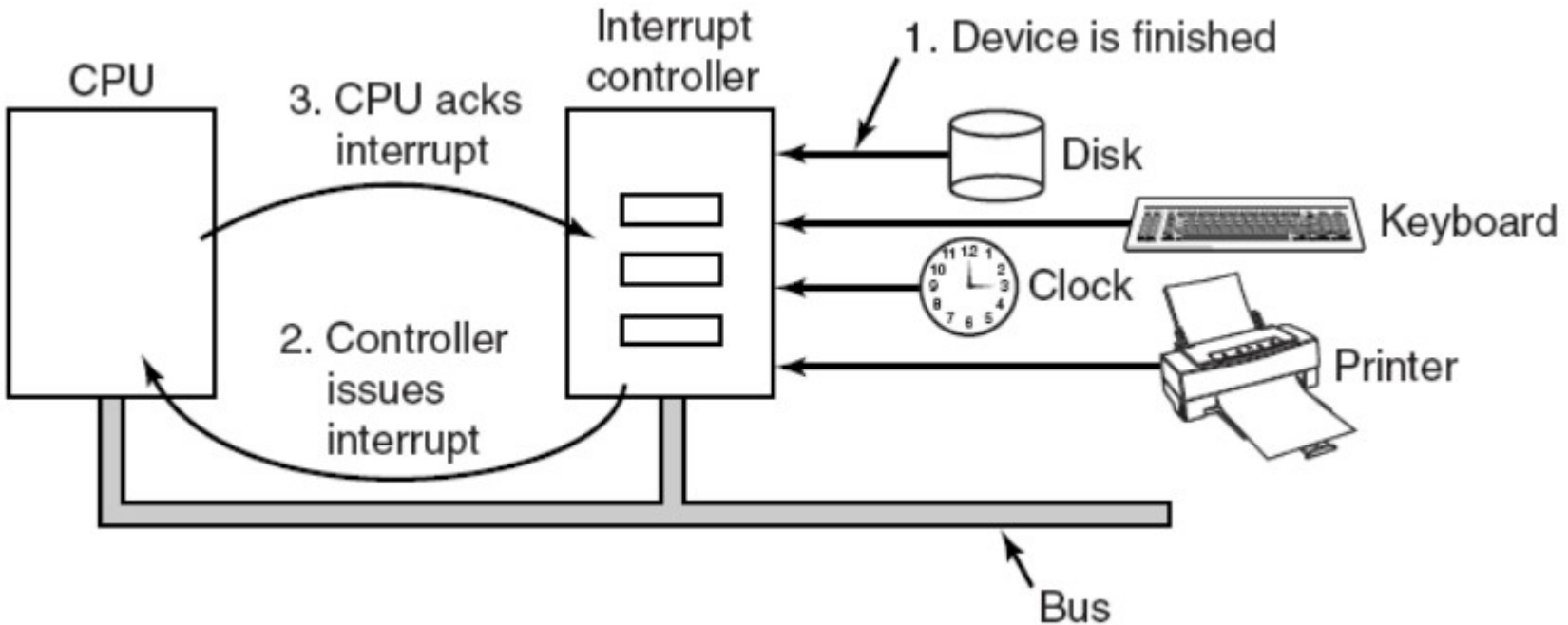
# *Interrupções*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - **Após o início da execução**, o **procedimento** de **serviço** de **interrupção reconhece** a **interrupção gravando** um **certo valor** em **uma** das **portas** de **E/S** do **controlador** de **interrupção**;
  - Esse **reconhecimento informa** ao **controlador** que é **livre** emitir **outra interrupção**;
  - Ao fazer com que a **CPU atrase** esse **reconhecimento até** que esteja **pronta** para **lidar** com a **próxima interrupção**, **condições** de **corrida** envolvendo múltiplas interrupções (quase simultâneas) **podem ser evitadas**;
  - O **hardware** sempre **salva** certas **informações** antes de iniciar o procedimento de serviço. **Quais** informações são salvas e **onde** são salvas **variam** muito de CPU para CPU. **No mínimo, o contador do programa deve ser salvo**, para que o processo interrompido possa ser reiniciado. **No outro extremo**, **todos** os **registradores visíveis** e um grande número de **registradores internos também** podem ser salvos;
  - Normalmente, os dados são salvos na pilha do kernel.

# *Interrupções*

- **Conceitos**



# *Interrupções*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Interrupções precisas e imprecisas**
  - As **CPUs** modernas utilizam **arquiteturas *pipeline*** e **superescalares**;
  - **Problema**: ocorre uma interrupção e várias instruções estão em processo (não terminadas). O que fazer?
  - Utilizar **interrupções precisas –** simplifica o tratamento da interrupção e uso da pilha,

# *Interrupções*

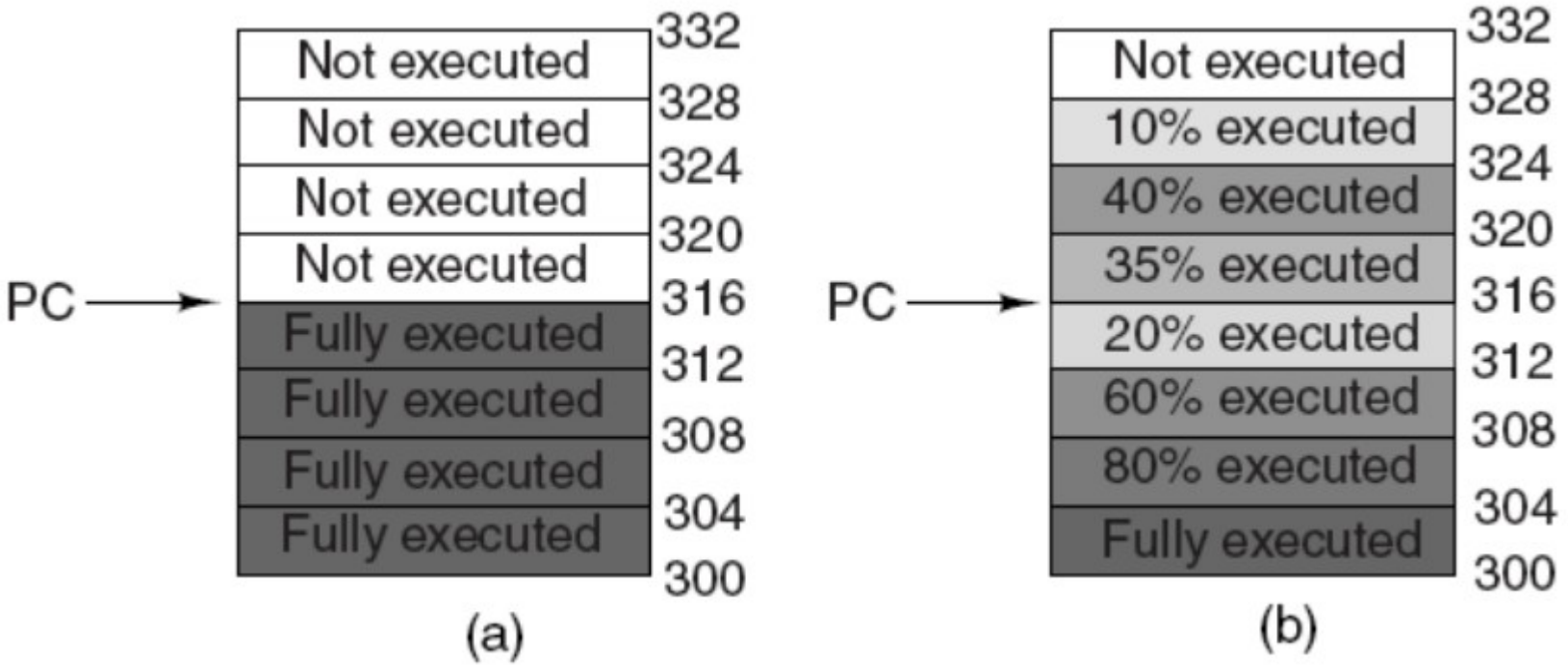
INSTITUTO MAUÁ DE TECNOLOGIA

- **Interrupções precisas e imprecisas**
  - **Características da interrupções precisas**
    1) O PC (contador de programas) é salvo em um local conhecido;
    2) Todas as instruções anteriores à apontada pelo PC foram totalmente executadas;
    3) Nenhuma instrução além da apontada pelo PC foi executada;
    4) O estado da execução da instrução apontada pelo PC é conhecido.

# *Interrupções*

- **Diferença entre interrupções precisas e imprecisas**

  (a) Interrupção precisa; (b) Interrupção imprecisa



# *Princípios de software de E/S*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Princípios**
  - **Independência de dispositivo**: os programas podem **acessar qualquer dispositivo** de **E/S** sem **saber** de **antemão detalhes intrínsecos** desse dispositivo;
  - **Nomeação uniforme:** o **nome** do **arquivo** ou **dispositivo** deve ser uma **cadeia** de **caracteres** ou **número** e não depender do dispositivo;
  - **Tratamento de erros**: **erros** devem ser **tratados próximos** ao **hardware** e **corrigidos** pelo seu controlador, **se possível**. **Senão**, devem ser **tratados** pelo **driver** do **dispositivo**;
  - **Acesso síncrono e assíncrono**: programação de E/S é mais simples quando se utiliza acesso síncrono (bloqueante), mas acesso assíncrono é mais eficiente;
  - **Buferização**: **necessário** em alguns **dispositivos**, como **interface** de **rede** (que não sabe para qual programa os dados serão utilizados);
  - **Compartilhamento de dispositivos**: em discos, o compartilhamento em leitura e escrita deve ser obrigatório, mas em impressoras não (impressão de um usuário por vez).

# *Entrada e Saída Programada*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - Consiste em **efetuar** uma **chamada** de **sistema** que se **encarrega** de realizar as **operações** de E/S;
  - Por **exemplo**, quando um **processo** do **usuário** que **imprimir** uma **cadeia** como “ABCDEFGH” via **interface serial**, o **software primeiro armazena** a **cadeia** em um **buffer** no espaço do usuário;
  - Depois, o **processo acessa** a **impressora** por meio de uma **chamada** de **sistema** para **abri-lo** Se a **impressora** está em **uso** por **outro processo**, esta **chamada falha** e **retorna** um **erro ou** então **bloqueia** até que a **impressora** esteja **disponível**;
  - Quando a **impressora** está **disponível**, o **processo** do usuário **executa** a **chamada** de **sistema** pedindo ao **sistema** operacional para **imprimir** a **cadeia** na impressora.

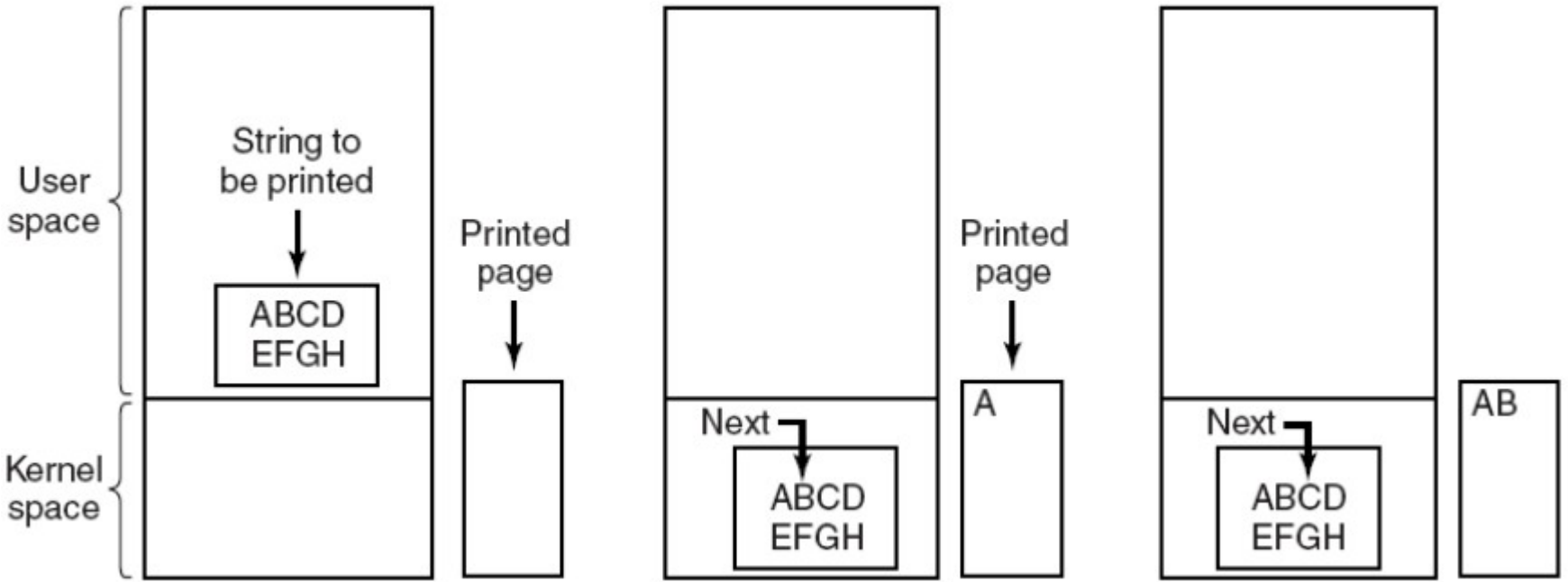
# *Entrada e Saída Programada*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - O **sistema operacional copia** o **buffer** em um **vetor** no **espaço** do **kernel**, **verifica** se a **impressora** está **disponível**, aguardando, caso contrário;
  - Quando a **impressora** está **livre**, o **sistema operacional copia** o **primeiro caractere** ao **registrador** de dados da **impressora** (E/S mapeada em memória) – um buffer – , **ativando** a **impressora** que a seguir se encarrega da **impressão**;
  - **Repete-se** este processo até que **toda** a **cadeia** seja **impressa**;
  - **Vantagem**: simplicidade;
  - **Desvantagem**: bloqueia a CPU até o serviço terminar.

# *Entrada e Saída Programada*

- **Conceitos**
  - Passos para imprimir uma cadeia de caracteres



# *Entrada e Saída Programada*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - **Chamada** de **sistema** para imprimir uma cadeia de caracteres

```
copy_from_user(buffer, p, count);                    /* p is the kernel buffer */
for (i = 0; i < count; i++) {                        /* loop on every character */
        while (*printer_status_reg != READY) ;    /* loop until ready */
        *printer_data_register = p[i];               /* output one character */
}
return_to_user( );
```

# *Entrada e Saída dirigida por Interrupção*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - Se no **exemplo** da **impressora** ela **não** tivesse um **buffer**, o **tempo ocioso** de CPU seria maior ainda;
  - Uma **forma** de **reduzir** este **problema** é **utilizar interrupções** assim:
    - Quando é **realizada** a **chamada** de **sistema** de impressão, o **primeiro caractere** é enviado para impressão;
    - Depois, o **agendador entra** em **ação** e pode **chamar outro processo**, **evitando** que a **CPU** fique **bloqueada** (mas o processo que está imprimindo fica);
    - Quando a **impressora termina** a **impressão**, ela **gera** uma **interrupção** que **faz** com que o **agendador retorne** ao **processo** de **impressão**, que repete o os passos anteriores até o fim da impressão.

# *Entrada e Saída dirigida por Interrupção*

- **Conceitos**

  (a) Código que é executado pela chamada de sistema `print()`; (b) ISP (*Interrupt Service Procedure*) associado à impressora.

```
copy_from_user(buffer, p, count);
enable_interrupts( );
while (*printer_status_reg != READY) ;
*printer_data_register = p[0];
scheduler( );
```

(a)

```
if (count == 0) {
    unblock_user( );
} else {
    *printer_data_register = p[i];
    count = count – 1;
    i = i + 1;
}
acknowledge_interrupt( );
return_from_interrupt( );
```

(b)

# *Entrada e Saída usando DMA*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**
  - No **exemplo** da **impressora**, a grande **desvantagem** de **E/S** usando **interrupção** é que a cada **caractere impresso**, uma **interrupção** é **gerada** e isso **toma tempo**;
  - A **solução** é utilizar **DMA**, deixando o **controlador** de **DMA alimentar** os **caracteres** para **impressora**, um por vez, sem passar pela CPU.

# *Entrada e Saída usando DMA*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Conceitos**

  (a) Código executado quando a chamada de sistema é realizada.
  (b) Procedimento de serviço de interrupção

```
copy_from_user(buffer, p, count);
set_up_DMA_controller( );
scheduler( );
```

(a)

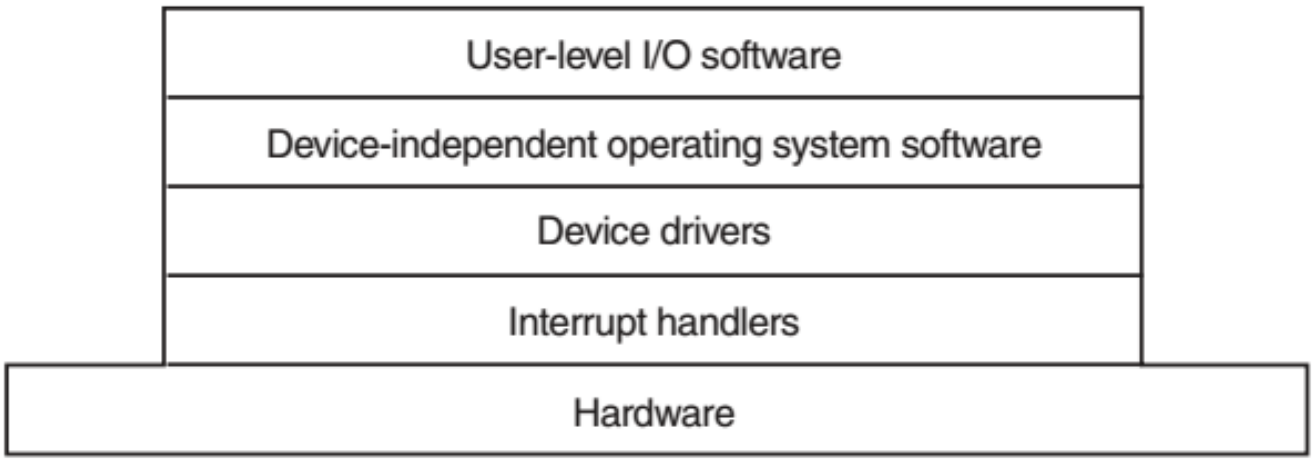
```
acknowledge_interrupt( );
unblock_user( );
return_from_interrupt( );
```

(b)

# *Camadas de software de E/S*

INSTITUTO MAUÁ DE TECNOLOGIA

- **Camadas**



# *Referências bibliográficas*

TANENBAUM, Andrew S. **Sistemas operacionais modernos**. 3. ed. São Paulo: Pearson, 2013. 653 p.