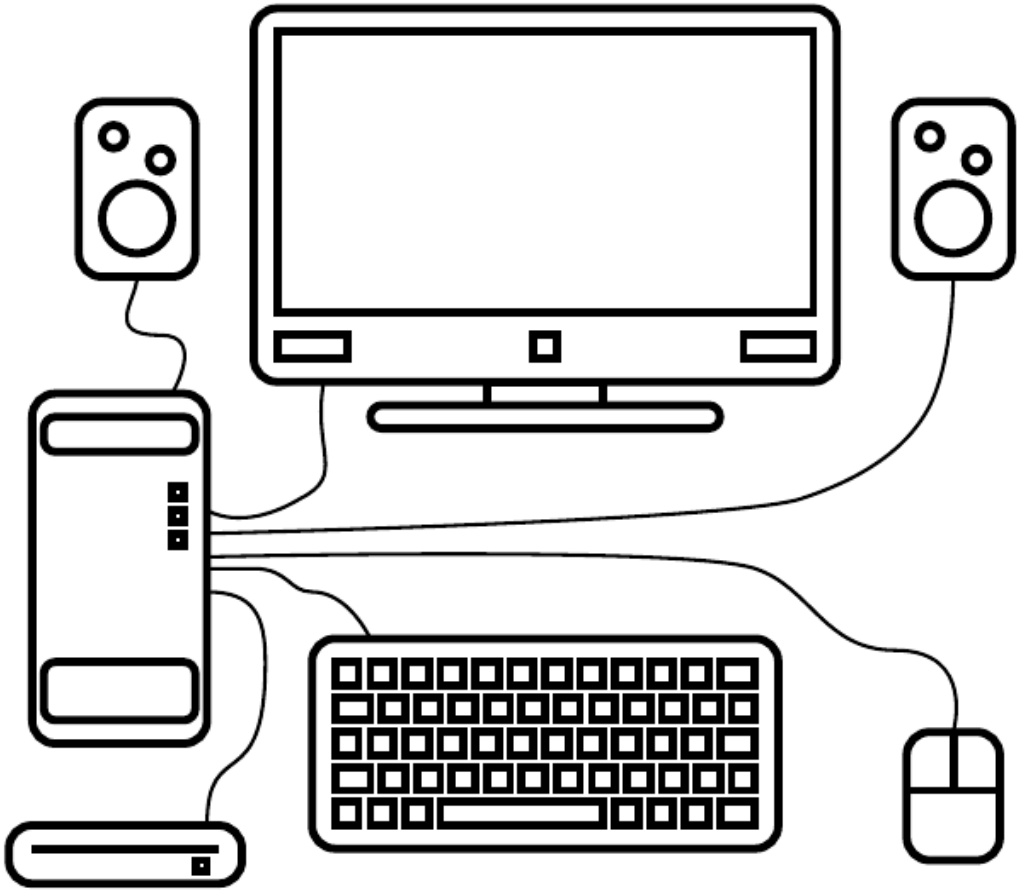
UNIVERSIDADE FEDERAL DO CEARÁ
Campus Russas
Disciplina: Sistemas Operacionais

Cap. 7 – Entrada/Saída e Impasses

Prof. Rafael Ivo

Introdução

- Dispositivos de E/S permitem a interação do computador com o mundo exterior de várias formas:

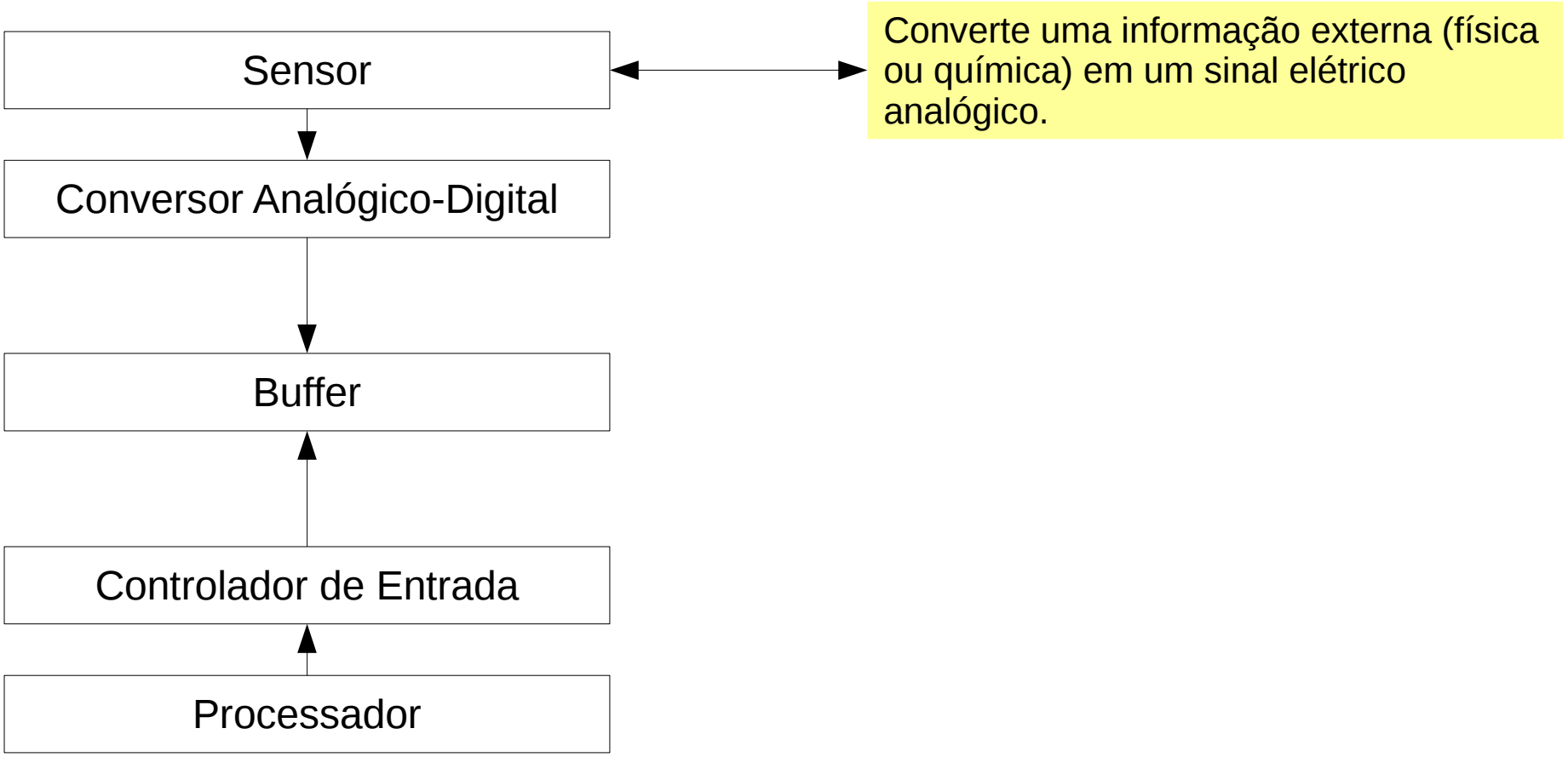


Introdução

- Dispositivos de E/S permitem a interação do computador com o mundo exterior de várias formas:
  - Interação com usuários através do *mouse, teclado, tela de toque, etc.*
  - Escrita e leitura de dados em discos rígidos, SSDs, CD-ROMs, pen-drives, etc.
  - Impressão de informações através de impressoras e plotadoras
  - Captura e reprodução de audio/vídeo
  - Comunicação entre outros computadores através de LAN, Bluetooth, telefonia celular, etc.

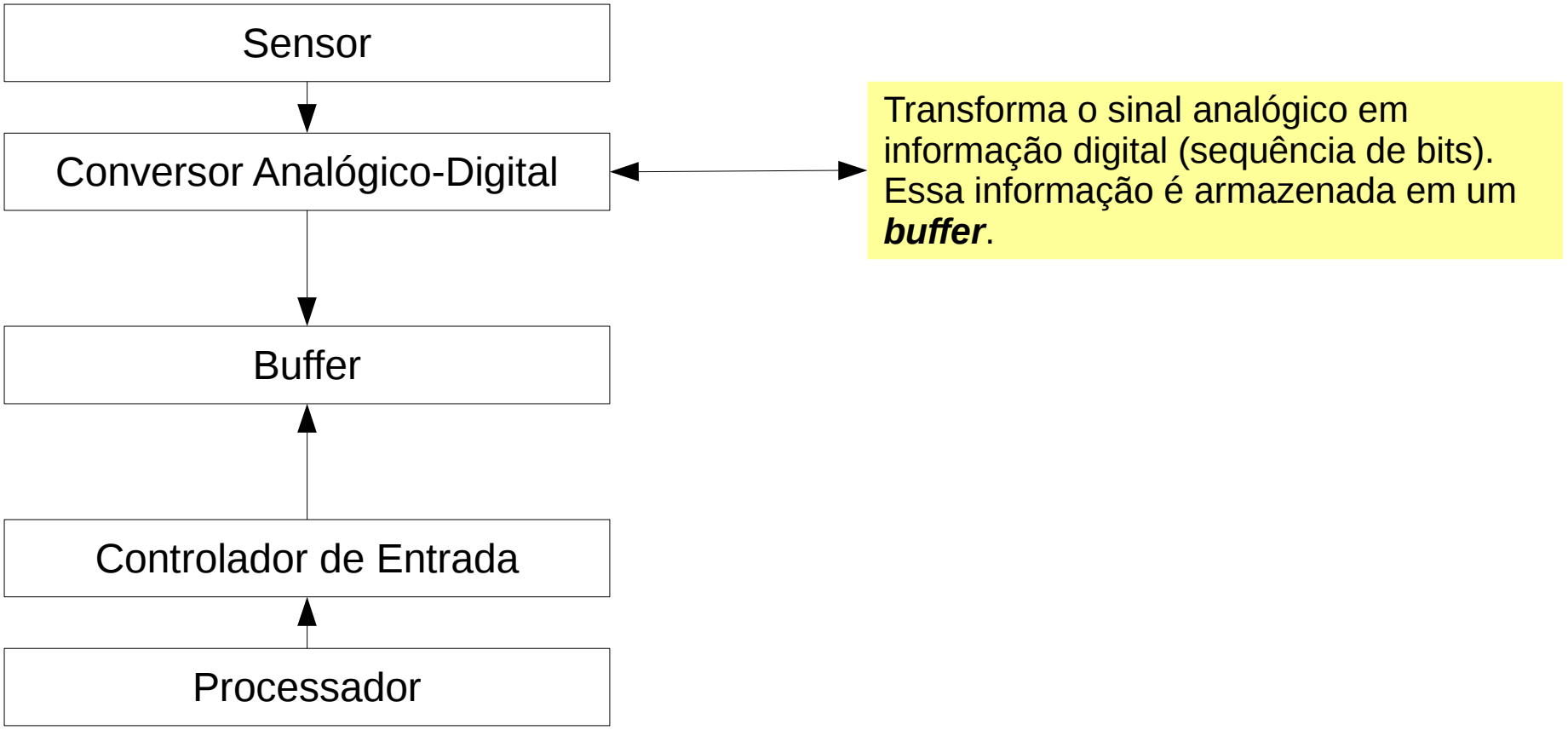
Introdução

- Conceitualmente, a entrada de dados em um computador é feita:



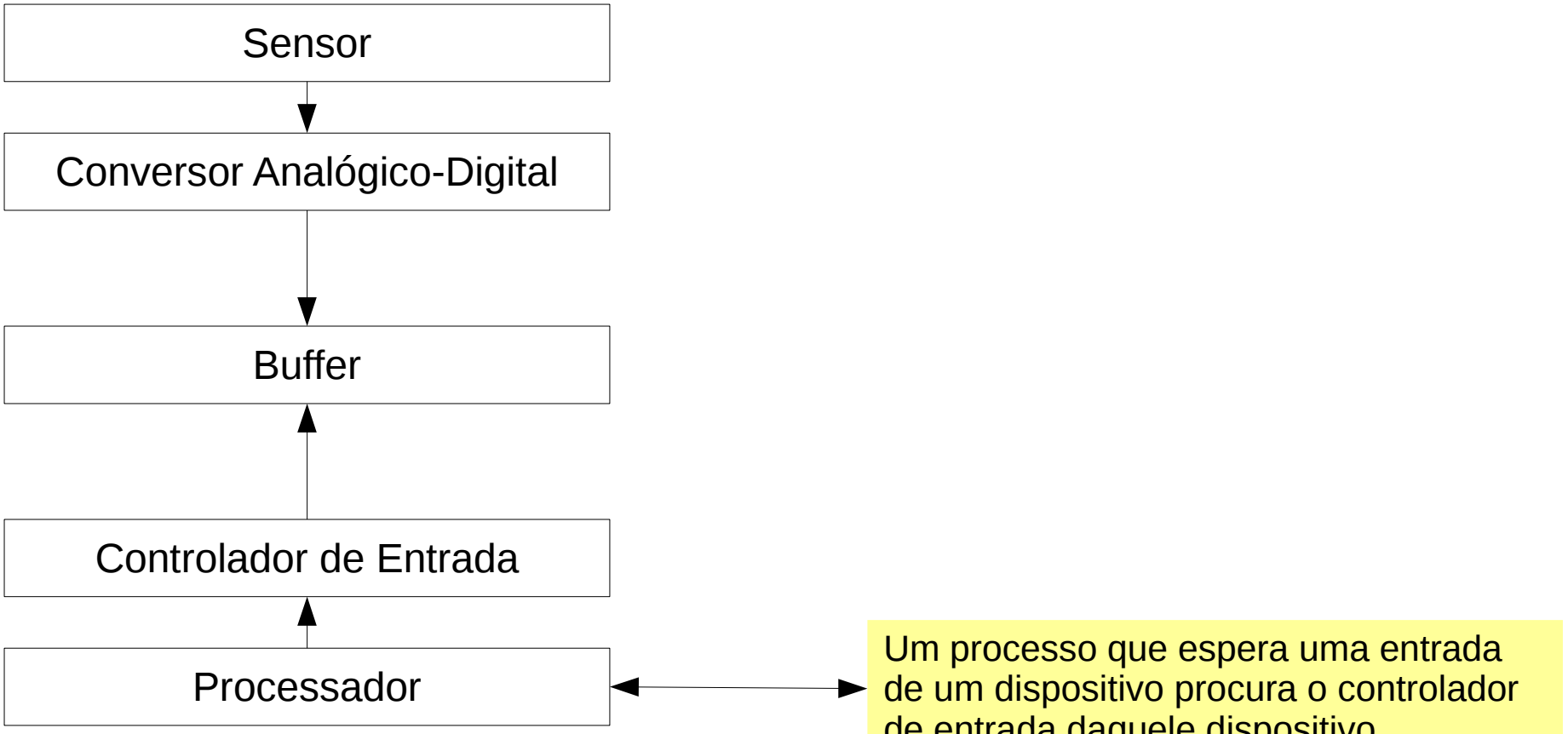
Introdução

- Conceitualmente, a entrada de dados em um computador é feita:



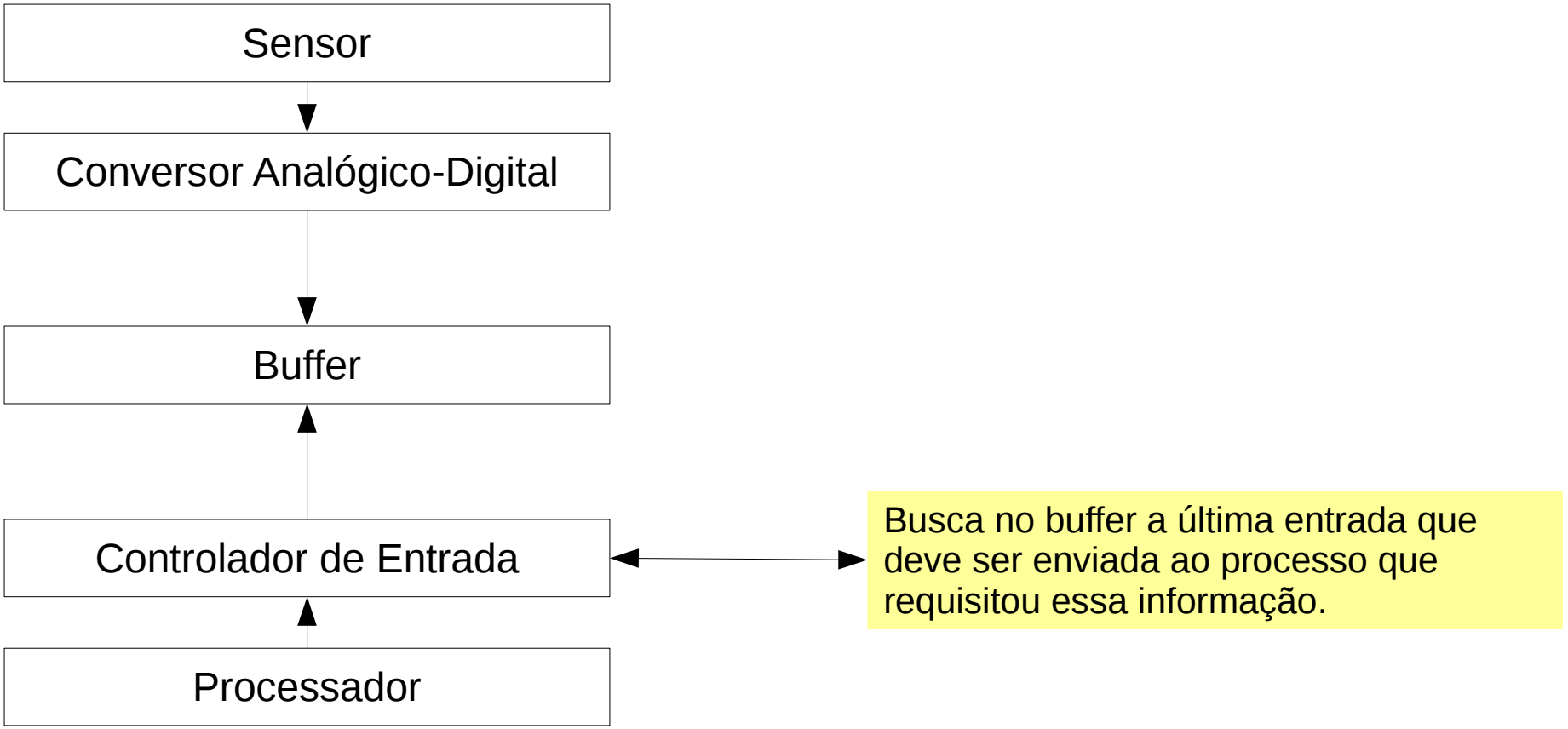
Introdução

- Conceitualmente, a entrada de dados em um computador é feita:



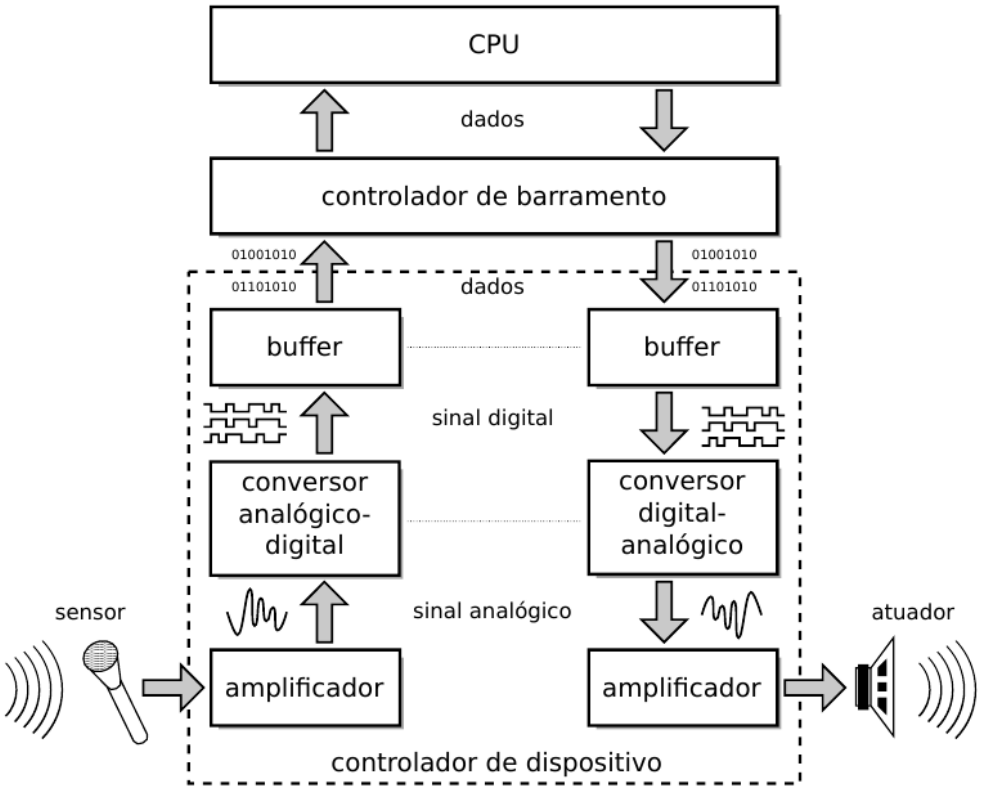
Introdução

- Conceitualmente, a entrada de dados em um computador é feita:



Introdução

- A saída de dados é feita de forma similar, mas seguindo o caminho contrário.
- Vários dispositivos combinam funcionalidades tanto de entrada quanto de saída
  - Ex: discos rígidos, placas de áudio, etc.



Introdução

- Dispositivos diferem em velocidade de transferência, forma de transferência dos dados e métodos de acesso

Dispositivo	**velocidade**
Teclado	10 B/s
Mouse ótico	100 B/s
Interface infravermelho (IrDA-SIR)	14 KB/s
Interface paralela padrão	125 KB/s
Interface de áudio digital S/PDIF	384 KB/s
Interface de rede *Fast Ethernet*	11.6 MB/s
Chave ou disco USB 2.0	60 MB/s
Interface de rede *Gigabit Ethernet*	116 MB/s
Disco rígido SATA 2	300 MB/s
Interface gráfica *high-end*	4.2 GB/s

Classes de Dispositivos

- Para simplificar as construções de aplicações (e do próprio SO), os dispositivos de E/S são agrupados em classes
- A classificação mais comum é a utilizada por Tannenbaum:
  - ***Dispositivos orientados a blocos***
    - Operações de E/S são feitas usando blocos de bytes e nunca bytes isolados
    - Ex: discos rígidos, fitas magnéticas, pen drives e outros dispositivos de armazenamento
  - ***Dispositivos orientados a caracteres***
    - Operações de E/S são feitas usando byte a byte ou usando blocos de bytes de tamanho variável, cujo tamanho mínimo é um byte
    - Ex: mouse, teclado, modems (linha telefônica)
  - ***Interfaces de rede (Exceção)***
    - Dispositivos orientados a blocos (os pacotes de rede) endereçáveis, mas a saída é feita de forma sequencial, bloco após bloco, e normalmente não é possível resgatar ou apagar um bloco enviado ao dispositivo
    - Ex: placa de rede

Hardware de E/S

- Interrupções
  - O acesso aos controladores dos dispositivos através de seus registradores é conveniente para a comunicação no sentido processador → controlador
  - Mas é problemática no sentido controlador → processador
    - Caso o controlador precise informar algo ao processador sem que ele esteja esperando
  - Neste caso, o controlador pode utilizar uma requisição de interrupção (IRQ = *Interruption Request*)
    - Notifica-se o processador sobre algum evento importante, como:
      - A conclusão de uma operação solicitada
      - A disponibilidade de um novo dado
      - A ocorrência de algum problema no dispositivo

Hardware de E/S

- Interrupções
  - Cada interrupção está associada a um número inteiro que permite identificar o dispositivo que a solicitou
  - Ex:

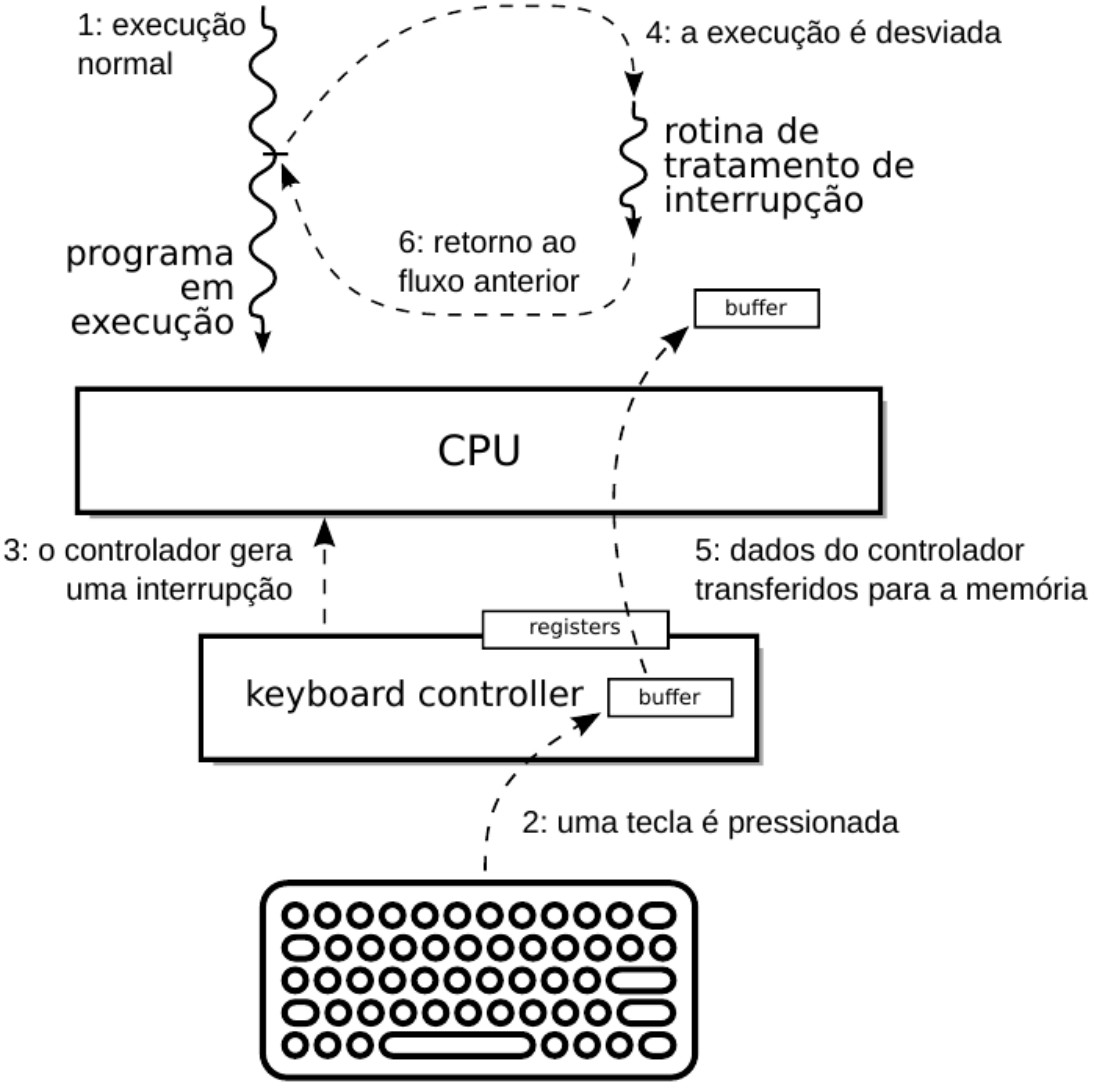
Dispositivo	Interrupção
teclado	1
mouse PS/2	12
barramento IDE primário	14
barramento IDE secundário	15
relógio de tempo real	8
porta serial COM1	4
porta serial COM2	3
porta paralela LPT1	7

Hardware de E/S

- Interrupções
  - Ao receber uma requisição de interrupção
    - O processador suspende seu fluxo de instruções corrente
    - Desvia a execução para um endereço pré-definido onde se encontra uma ***rotina de tratamento de interupção*** (*interrupt handler*)
    - Depois da execução dessa rotina, o processador retoma o código que estava executando quando foi interrompido

Hardware de E/S

- Interrupções
  - Ex:



Hardware de E/S

- Interrupções
  - Como cada interrupção corresponde a um evento ocorrido em um dispositivo periférico
    - Chegada de um pacote de rede
    - Movimento do mouse
    - Conclusão de operação do disco
    - Etc.
  - … podem ocorrer centenas ou milhares de interrupções por segundo
  - Por isso, as rotinas de tratamento de interrupções devem realizar suas tarefas rapidamente para não prejudicar o funcionamento do restante do sistema.

Hardware de E/S

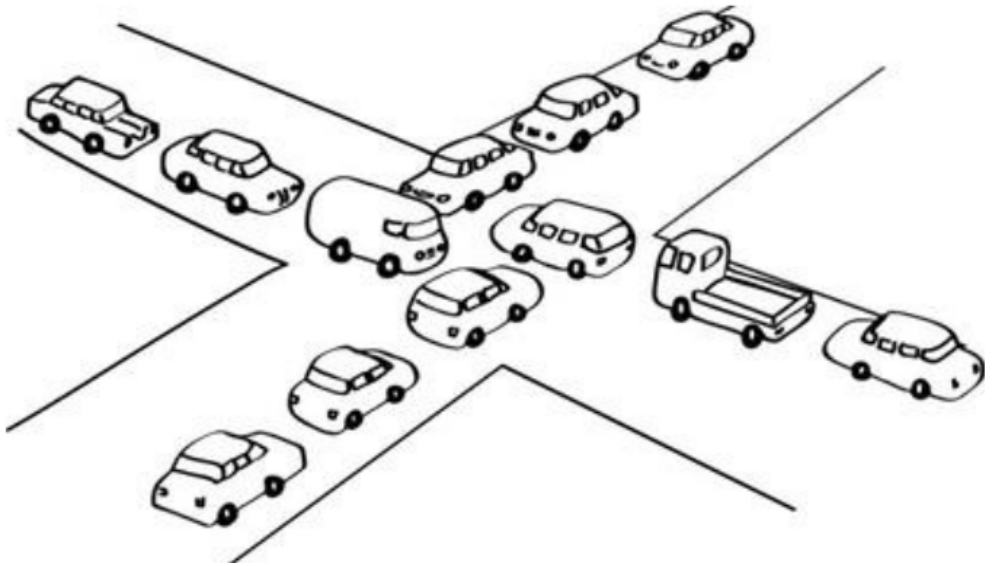
- Interrupções
  - Não existe uma única rotina para tratamento de interrupção para todos os dispositivos de E/S
  - A maioria das arquiteturas atuais define um vetor de endereços de funções denominado ***vetor de interrupções*** (IV – *Interrupt Vector*)
    - Cada entrada desse vetor aponta para a rotina de tratamento da interrupção correspondente
    - Ex: se o teclado (código 1) emitir o pedido de interrupção, o vetor de interrupções na posição 1 conterá o endereço onde tem a implementação do tratamento de teclado

Hardware de E/S

- Interrupções
  - Conclusão
    - O mecanismo de interrupção torna eficiente a interação do processador com os dispositivos periféricos
    - Se não existisse as interrupções, o processador teria que regularmente ficar verificando se há eventos a serem tratados
    - O processador não precisa esperar a conclusão de cada operação solicitada, pois o dispositivo emitirá uma interrupção para “avisar” o processador quando a operação for concluída

Impasses

- Significado do Dicionário
  - *“Situação cuja saída ou resolução é praticamente impossível ou muito difícil”*
- Vejamos o seguinte problema do cotidiano de uma cidade grande:



- Cada carro precisa de estrada livre a sua frente para se mover.
- Cada carro ocupa uma área da estrada.
- Na situação mostrada na figura, cada carro no cruzamento deseja avançar a frente.
- Entretanto, o carro a sua frente não consegue se mover por esperar o carro na transversal.
- A dependência entre eles torna a fluidez do trânsito em um impasse, causando o congestionamento.

Impasses

- Em um sistema computacional, o problema pode ocorrer da seguinte forma. Por exemplo:
  - O processo A, que está usando a impressora, solicita o uso do scanner
  - O processo B, que está usando o scanner, solicita o uso da impressora
  - Ambos os processos serão bloqueados esperando um pelo outro e ficarão para sempre nesta situação
- O problema surge quando dois ou mais processos tentam acessar recursos exclusivos um do outro.

Impasses

- Uma classe importante de impasses envolve recursos.
- Para tornar a discussão o mais geral possível, faremos referência aos objetos acessados como ***recursos***
  - Pode ser um dispositivo de hardware (ex: impressora)
  - Pode ser uma informação (ex: um registro de banco de dados)
- Para nossos estudos, classificaremos estes recursos em dois tipos:
  - Preemptivos
  - Não preemptivos

Impasses

- Recursos preemptivos
  - Podem ser interrompidos e retirados de um processo sem prejuízos
  - Ex: Memória Primária
    - Um processo que precisa de memória primária, mas não há mais disponível pode ser enviado ao disco e depois sua execução retomada, carregando-o para a memória em outra oportunidade.
    - Ou seja, o uso da memória por um processo pode ser interrompido momentaneamente sem prejuízo do processo.

Impasses

- Recursos não-preemptivos
  - Não pode ser retirado do processo sem potencialmente causar dano
  - Ex: Gravadora de CD
    - Retirar repentinamente de um processo que esteja usando o gravador de CD para passar o uso desse recurso a outro processo resultará em um CD com erros
  - ***Impasses ocorrem com esse tipo de recurso***

Impasses

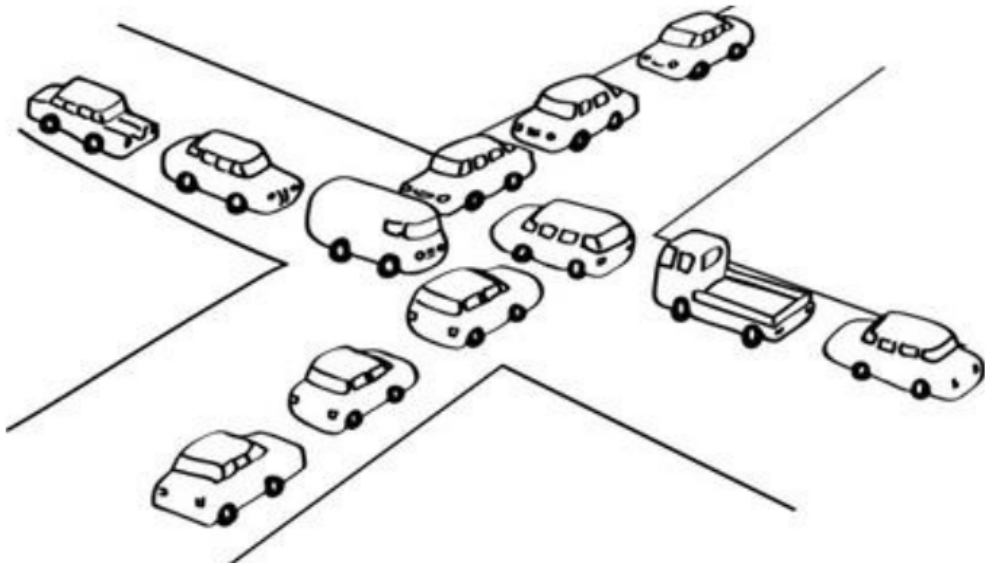
- Sequência de eventos para utilizar um recurso compartilhado
  - Requisição do recurso
  - Utilização do recurso
  - Liberação do recurso
- Se não estiver disponível, o que pode ocorrer?
  - Processo que requisitou o recurso fica bloqueado esperando o recurso ser liberado
  - Processo espera um certo tempo e volta a requisitar o recurso novamente

Impasses

- Há 4 condições são necessárias para que um impasse (deadlock) aconteça
  - ***Exclusão mútua***
    - Um determinado recurso só pode estar sendo usado por um único processo em um determinado instante. Ou ele não está sendo usado por nenhum processo e está disponível.
  - ***Uso e Espera***
    - Processos que já possuem algum recurso podem pedir por outros recursos para finalizar
  - ***Não-preempção***
    - Recursos já alocados não podem ser retirados do processo que os alocou. Somente o próprio processo que alocou pode liberar o recurso.
  - ***Dependência circular***
    - Há um encadeamento circular de dois ou mais processos; cada um esperando recursos que estão sendo usados pelo membro seguinte do ciclo.

Impasses

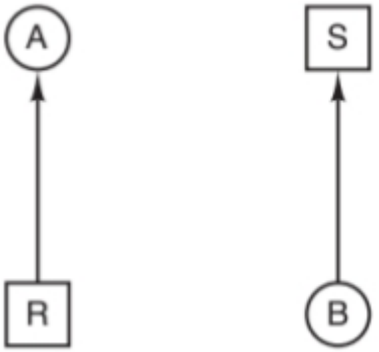
- Analogia com o congestionamento...



- ***Exclusão mútua***
  - Cada carro ocupa um pedaço da rua (recurso) e nenhum outro carro pode ocupar a mesma área ao mesmo tempo
- ***Posse e Espera***
  - Cada carro mantém seu “pedaço” de rua, mas deseja o “pedaço” da frente para poder se deslocar.
- ***Não-preempção***
  - A área ocupada por um carro só pode ser liberada por ele próprio. Ninguém pode removê-lo contra a sua vontade.
- ***Espera circular***
  - Cada carro no cruzamento da pista espera obter o “pedaço” da rua a sua frente que está ocupado por um outro carro também esperando para obter o pedaço a frente dele. Essa dependência forma um ciclo.

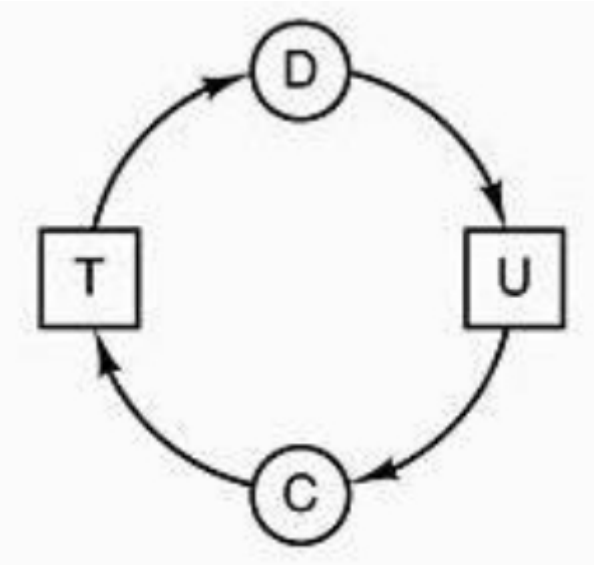
Impasses

- Modelagem de Deadlocks
  - As condições podem ser visualizadas através de ***grafos direcionados***
    - Processsos são círculos
    - Recursos são quadrados
    - Quando um recurso está alocado a um processo, há uma aresta saindo do recurso para o processo
    - Quando um processo está solicitando um recurso, há uma aresta saindo do processo para o recurso
  - Ex:
    - Recurso R está alocado ao processo A
    - Processo B solicita o recurso S



Impasses

- Modelagem de Deadlocks
  - Dependências circulares são detectadas neste grafo ao encontrar ciclos
  - Ex:
    - Recurso U alocado ao processo C
    - Recurso T alocado ao processo D
    - Processo C solicita recurso T
    - Processo D solicita recurso U ***(Fecha o ciclo → Deadlock!)***



Impasses

- Estratégias para tratar Deadlocks
  - Ignorar por completo o problema
  - Detecção e recuperação
  - Alocar cuidadosamente os recursos
  - Prevenir evitando uma das 4 condições necessárias

Impasses

- Ignorar o problema
  - Algoritmo do Avestruz: “Finja que nada está acontecendo”

Impasses

- Ignorar o problema
  - Por que esta “solução” é cogitada?
  - Deve-se fazer as seguintes perguntas:
    - Deadlocks são frequentes?
    - Vale a pena pagar o preço em termos de desempenho?
  - Maioria dos S.O. (Windows e Unix) ignora o problema supondo que a maior parte dos usuários preferiria um deadlock ocasional a alguma regra que restrinja a usabilidade do sistema.

Impasses

- Detecção e Recuperação
  - Deadlocks podem acontecer, mas o sistema tenta detectar as causas e solucionar a situação
  - A detecção consiste em analisar o grafo mostrado anteriormente em busca de ciclos
  - As recuperações possíveis são:
    - ***Por meio de preempção***
      - Retirar o recurso de um processo e dar a outro.
      - Não se pode fazer com qualquer recurso
    - ***Por meio de reversão de estado (volta ao passado)***
      - Utiliza-se checkpoints, gravando o estado do processo
      - Ao detectar o deadlock, tenta retornar um dos processos ao estado antes de acontecer o deadlock
      - Difícil de ser implementado e gasta bastante desempenho computacional
    - ***Por meio de eliminação de processo***
      - Mata um processo do ciclo liberando os demais a continuarem sua execução
      - Solução mais simples e eficiente

Impasses

- Prevenção
  - Se evitarmos que uma das 4 condições ocorra, evita-se o deadlock de acontecer
  - Atacando a condição da ***exclusão mútua***
    - Alguns recursos podem ser tratados por um gerente que mantém uma fila dos processos (***spool***) que pediram para usar o dispositivo
    - Somente o gerente utiliza o dispositivo
    - Elimina-se o ciclo, eliminando a possibilidade de deadlock envolvendo o recurso
    - Problema: nem todo dispositivo pode ser tratado desta forma

Impasses

- Prevenção
  - Se evitarmos que uma das 4 condições ocorra, evita-se o deadlock de acontecer
  - Atacando a condição da ***Posse e Espera***
    - Exigir que todos os processos requisitem todos os recursos de uma única vez
    - Obtém tudo ou nada
    - Sem “segurar” um recurso enquanto espera por outro, não forma-se ciclos
    - Problema: Um processo não tem conhecimento dos recursos que ele precisará antes de começar

Impasses

- Prevenção
  - Se evitarmos que uma das 4 condições ocorra, evita-se o deadlock de acontecer
  - Atacando a condição da ***Não-preempção***
    - Retira-se o recurso de algum processo para quebrar o ciclo
    - Problema: solução inviável para alguns dispositivos como mostrado anteriormente

Impasses

- Prevenção
  - Se evitarmos que uma das 4 condições ocorra, evita-se o deadlock de acontecer
  - Atacando a condição da ***Dependência Circular***
    - Atribui um código para cada recurso
    - Exige que cada processo só possa pedir recursos em ordem crescente de enumeração
    - Solução mais atrativa de ser praticada.

Impasses

- Prevenção
  - Se evitarmos que uma das 4 condições ocorra, evita-se o deadlock de acontecer
  - Resumo

Condição	Abordagem contra impasses
Exclusão mútua	Usar spool em tudo
Posse e espera	Requisitar todos os recursos necessários no início
Não preempção	Retomar os recursos alocados
Espera circular	Ordenar numericamente os recursos