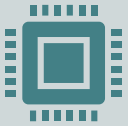
Instruções, Interrupções e Exceções

Relembrando...

Estrutura Básica do Computador

- O computador é composto, basicamente, por:
  - CPU
  - Memória
  - Dispositivos de entrada / saída

- A CPU é o principal elemento do computador, responsável pela realização das operações de processamento e de controle durante a execução dos programas.



Para que um programa seja executado pela CPU, é necessária a execução de diversas instruções, que estão em linguagem de máquina.

Cada uma dessas instruções é armazenada em posições na memória do computador e são executadas uma a uma.

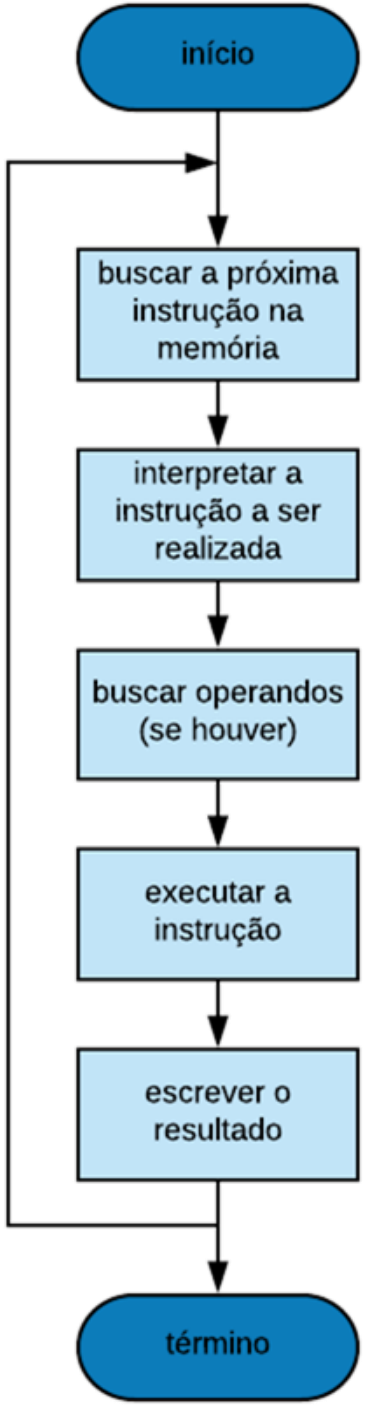
A execução é sequencial, ou seja, se a instrução executada está na posição x, a próxima instrução a ser executada deverá estar na posição x + 1.

Cada instrução atuará sobre um ou mais dados que são o(s) operando(s) da instrução, gerando um resultado.

Exemplo:

- A instrução "soma" requer dois números como operandos, gerando um terceiro número como resultado. Os dados processados no programa também ficam armazenados na memória.

A sequência de funcionamento de uma CPU é conhecida como ciclo de busca e execução de instruções, ou "busca – decodificação – execução" de instruções.


início
buscar a próxima instrução na memória
interpretar a instrução a ser realizada
buscar operandos (se houver)
executar a instrução
escrever o resultado
término

- As atividades realizadas pela CPU podem ser:
  - **Funções de processamento:** realiza atividades relacionadas com a execução de uma operação, ou seja, processa as instruções. **O componente que faz isso é a ULA (unidade lógica aritmética)** e ela é complementada pelos registradores.
  - **Funções de controle:** são realizadas pelos componentes da CPU que se encarregam de atividades de busca, interpretação e controle da execução das instruções, bem como do controle da ação dos demais componentes do sistema de computação (memória, entrada/saída).
    - **O componente que faz isso é a UC (unidade de controle)** e ela executa o ciclo de busca e execução gerando os sinais adequados para os demais componentes da CPU e do computador.

Instruções

- Nos sistemas atuais, chamados de multiprogramados, vários programas são executados simultaneamente pela CPU.
- Assim, várias instruções são executadas simultaneamente.
- A utilização concorrente da CPU deve ser implementada de maneira que, quando um programa perde o uso do processador e depois retorna para continuar o processamento seu estado deve ser idêntico ao do momento em que foi interrompido.

- O programa deverá continuar sua execução exatamente na instrução seguinte àquela em que havia sido interrompido.

- Para o usuário este processo é transparente, ficando ao usuário a percepção de que o computador está totalmente dedicado ao que ele está fazendo e a mais nenhum outro processo.

- Este mecanismo de concorrência é que permite o compartilhamento de recursos pelos vários programas sendo executados pelo processador.

- Para que um programa seja executado pela CPU, é necessária a execução de diversas instruções, que estão em linguagem de máquina.
- Cada uma dessas instruções é armazenada em posições na memória do computador e são executadas uma a uma.
- A execução é sequencial, ou seja, se a instrução executada está na posição x, a próxima instrução a ser executada deverá estar na posição x + 1.
- Cada instrução atuará sobre um ou mais dados que são o(s) operando(s) da instrução, gerando um resultado.

- **Exemplo:**
  - A instrução "soma" requer dois números como operandos, gerando um terceiro número como resultado.
  - Os dados processados no programa também ficam armazenados na memória.
  - A sequência de funcionamento de uma CPU é conhecida como ciclo de busca e execução de instruções, ou "busca – decodificação – execução" de instruções.

Ciclo de Busca e Execução (ou Ciclo de Instrução)

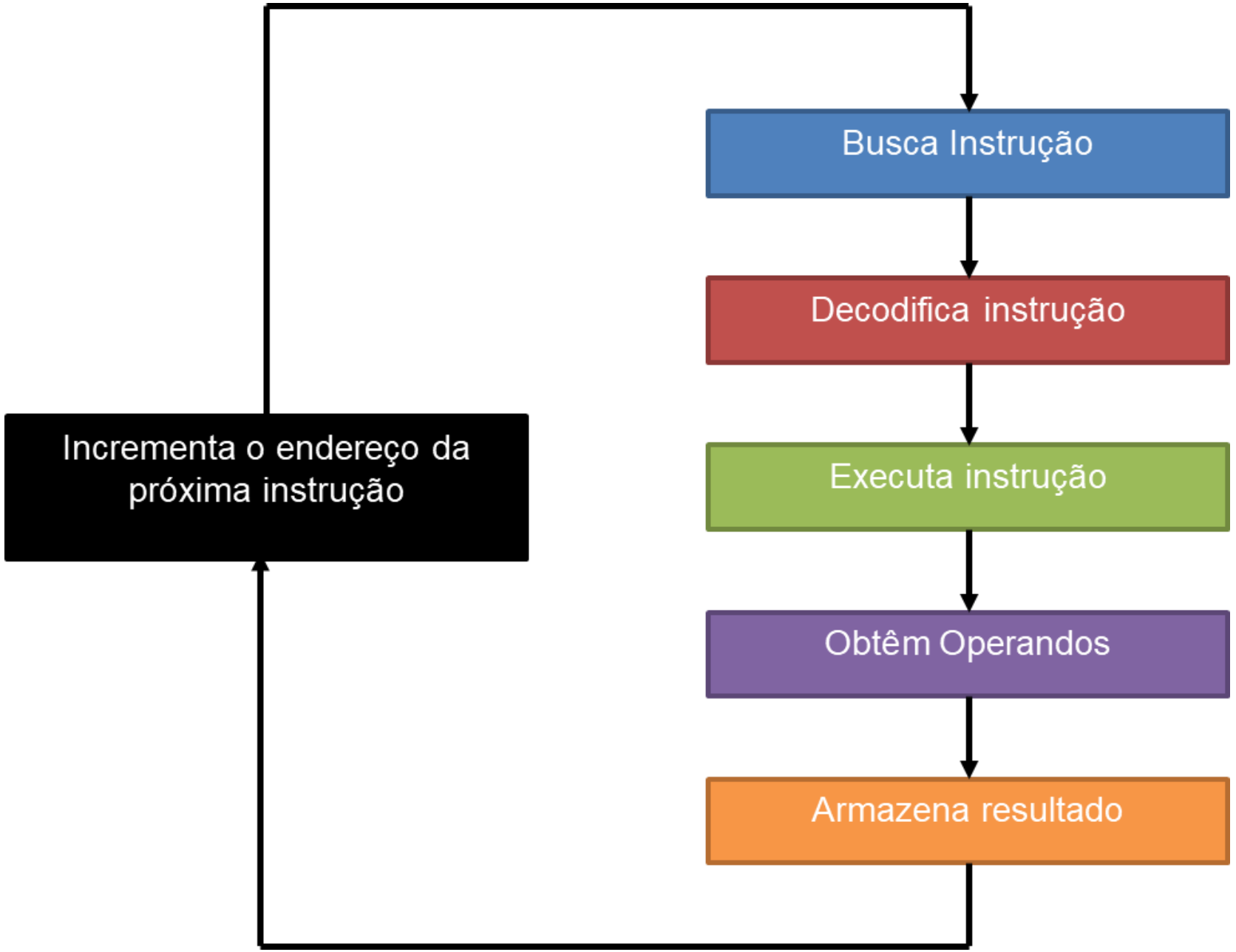
- O registrador Contador de Programa (PC – Program Counter) contém a posição da próxima instrução a ser executada.

- Quando uma sequência de execução de instruções tem início, a instrução cujo endereço está no Contador de Programa é trazida da memória para o Registrador de Instruções (RI). Esse processo é conhecido como busca da instrução.

- A próxima etapa corresponde à "interpretação" da instrução pelos circuitos da UC, ao que chamamos de decodificação da instrução.

- Conhecida a instrução, a etapa da execução da instrução envolve a obtenção dos operandos (dados manipulados pela instrução), efetivação da operação e armazenamento dos resultados.

- Quando a execução de uma instrução é terminada, o contador de instruções é atualizado para o endereço da memória da próxima instrução (x + 1).


Busca Instrução
Decodifica instrução
Executa instrução
Obtêm Operandos
Armazena resultado
Incrementa o endereço da próxima instrução

- A sequência de instruções pode mudar como resultado de uma instrução que direciona um desvio.
- Instruções desse tipo contêm o endereço da próxima instrução a ser executada.
- Elas causam mudanças no fluxo do programa dependendo do resultado do processamento.
- O desvio condicional representado por uma instrução de alto nível "IF" traduz-se em algum tipo de instrução de desvio.

- Processar dados é executar com eles uma ação que produza algum resultado. Essa é a função básica dos computadores.

- Dentre as tarefas de processamento de dados podem ser citadas:
  - **Operações aritméticas:** soma, subtração, multiplicação e divisão;
  - **Operações lógicas:** and, or, xor, not;
  - **Movimentação de dados:** memória – CPU (registrador), CPU (registrador) – memória, registrador – registrador;
  - **Desvios:** alteração da sequência de execução das instruções;
  - **Operações de entrada ou saída.**

Sistemas Multiprogramáveis

- Possibilidade de periféricos e dispositivos funcionarem simultaneamente junto com a CPU permitiu execução de tarefas concorrentes.
- Sistemas Operacionais podem ser vistos como um conjunto de rotinas que executam concorrentemente de uma forma ordenada
- Sistemas multiprogramáveis X baixa utilização dos recursos do sistema.

- Uso Médio CPU:
  - **Monoprogramáveis 30% X Multiprogramáveis 90%.**
- Vários programas podem estar residentes na memória, deixando-a menos ociosa
- Quando um programa perde o uso do processador, o estado do processamento deve ser armazenado para quando ele retornar para continuar executando a partir de onde parou.
- Compartilhamento de periféricos e recursos do sistema por vários usuários e programas.
- Maior complexibilidade do Sistema Operacional.
  - **O Kernel do SO é responsável por gerenciar recursos no SO tais como tratamento de interrupções e exceções, criação e eliminação de processos, e sincronização e comunicação.**

Interrupções

- Tornou possível a implementação da concorrência nos sistemas multiprogramáveis.
- Eventos que causam intervenção no Sistema Operacional durante a execução de programas.
- Gerados pelo próprio Sistema Operacional ou por Hardware.
- O sistema é desviado para uma rotina especial de tratamento.

- **Vetor de Interrupção** – Relação de todas as rotinas de tratamento das interrupções.
- **Mecanismo de Interrupção** – Procedimento para detectar a interrupção, salvar o contexto do programa e desviar para a rotina de tratamento. Na maioria das vezes implementados pelos projetistas e realizados pelo hardware.
- **Mascaráveis (podem ser desabilitadas) X Não-Mascaráveis (tratamento obrigatório).**
- As interrupções possuem **prioridades de execução.**
- **Controlador de pedidos de interrupção** – avalia as interrupções geradas e suas prioridades de atendimento.

- A interrupção é um mecanismo usado para fazer com que a CPU interrompa temporariamente a sequência de instruções (programa) em execução para atender um outro dispositivo que solicitou uma interrupção no processamento.

- A CPU não ficará "pausada" sem executar nada. O que ocorre é que a CPU passa a executar outra sequência de instruções (programa) para tratar a causa da interrupção, retornando posteriormente ao programa original.

- As interrupções foram criadas para evitar o desperdício do tempo computacional em loops de software enquanto esperam eventos que devem ser disparados por um determinado dispositivo.

- Os eventos que causam as interrupções podem ser:
  - **Internos** (ocorrência de overflow, tentativa de executar instrução não definida, chamada ao sistema)
  - **Externos** (pedidos dos dispositivos de E/S).

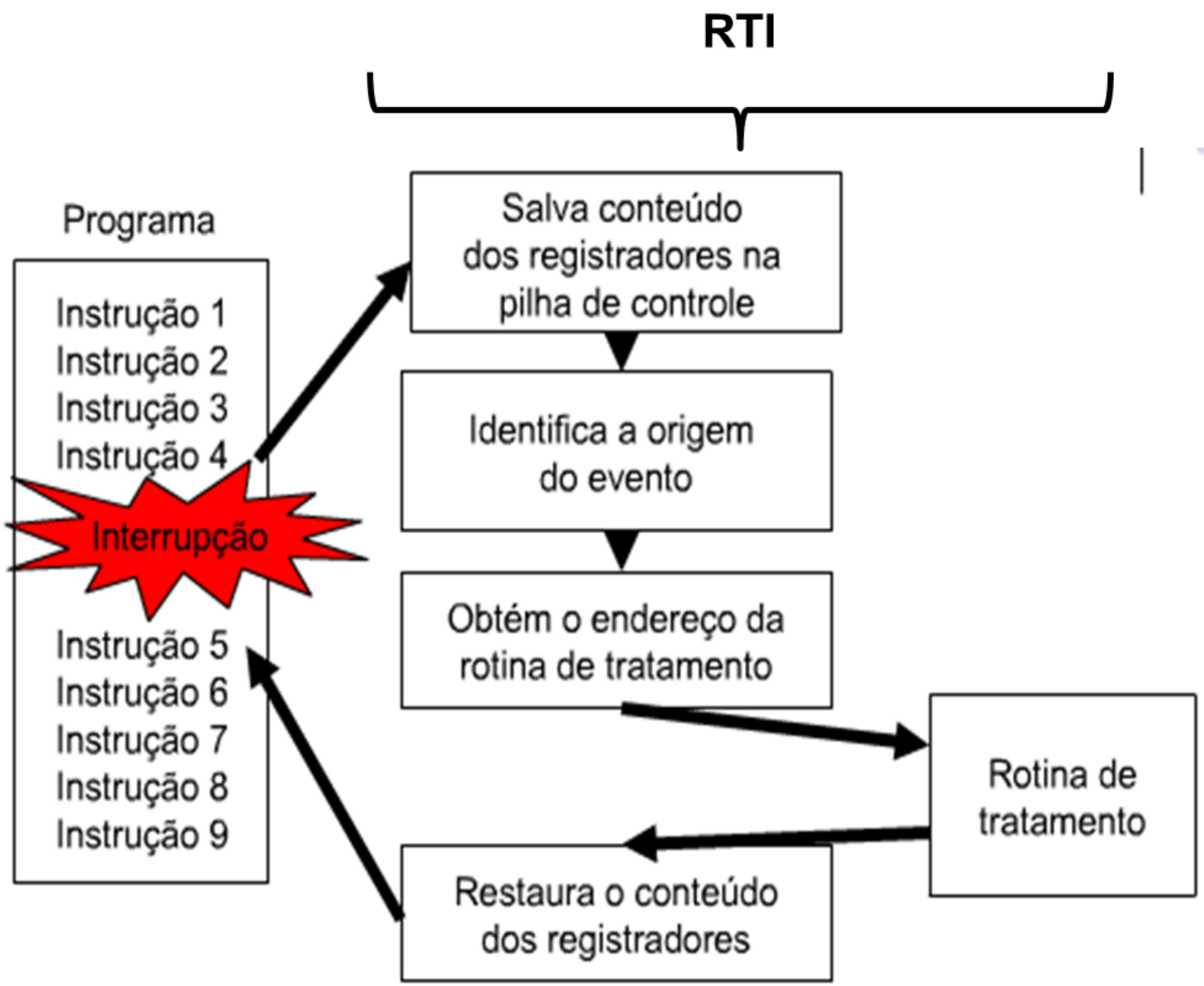
- **Existem quatro tipos (causas) de interrupções:**
  - **Temporização:** usada sempre que uma tarefa precisa de um tempo determinado; é uma espécie de temporizador que alerta quando o tempo programado é atingido. Ela é parte essencial na implementação da multitarefa pelo sistema operacional.
  - **Entrada e Saída (E/S)**: ocorre sempre que um dispositivo de E/S possui dados para a CPU, ou terminou uma operação solicitada por ela;
  - **Falha de software (exceção)**: acontece quando há uma falha no programa;
  - **Falha de hardware.**

- Fisicamente há um ou mais pinos do circuito integrado microprocessador dedicados à interrupção. A ocorrência de uma interrupção é indicada através de um sinal digital (bit) nesses pinos.

- A ocorrência de um interrupção é verificada ao final da execução de cada instrução de máquina (no final do ciclo de instrução).

- Se não houver nenhum pedido de interrupção, a CPU busca a próxima instrução normalmente.

- **Ciclo da interrupção:**
  - Interrupção é adicionada ao ciclo de instrução;
  - Processador verifica existência de interrupção;
  - Se não há interrupção, busca próxima instrução;
  - Se há interrupção:
    - CPU suspende execução do programa atual;
    - Salva o contexto atual (dados dos registradores da CPU);
    - Define registrador Contador de Programa (PC) para endereço inicial da Rotina de Tratamento de Interrupção (RTI);
    - Trata a interrupção (executa RTI);
    - Restaura o contexto;
    - Continua o programa interrompido.


RTI
Programa
Instrução 1
Instrução 2
Instrução 3
Instrução 4
Interrupção
Instrução 5
Instrução 6
Instrução 7
Instrução 8
Instrução 9
Salva conteúdo dos registradores na pilha de controle
Identifica a origem do evento
Obtém o endereço da rotina de tratamento
Rotina de tratamento
Restaura o conteúdo dos registradores

- Para que se possa retornar ao programa interrompido exatamente onde ele foi interrompido, é necessário salvar as informações presentes nos registradores (contexto).

- É possível que ocorra mais de um pedido de interrupção ao mesmo tempo.

- Para prevenir isso, o mais comum é que os sistemas estabeleçam alguns níveis de prioridade. Assim, as interrupções são atendidas "em fila" conforme a ordem de prioridade.

- Cada dispositivo possui um número de interrupção específico, que são as **IRQs (Interrupt Request ou Solicitações de interrupção)**, e essas **IRQs** são usadas para o processador identificar qual dispositivo está pedindo a interrupção.

- O número da **IRQ** indica, ainda, sua prioridade de execução, começando em 0, que é a prioridade mais alta.

Exceções

- As exceções são parecidas com as interrupções. Sua principal diferença é o motivo pelo qual o evento é gerado.

- A exceção é o resultado direto de uma instrução dentro do próprio programa, como a divisão por zero ou a ocorrência de um erro de estouro de memória em uma operação aritmética (overflow).

- As exceções são, portanto, geradas por eventos síncronos (que ocorrem ao mesmo tempo) dentro do próprio programa.
- Assim, tais eventos são previsíveis e podem ser tratados pelo programador que pode incluir tratamento de exceção no programa para que o mesmo não seja interrompido.
- O tratamento de exceção é similar ao tratamento de interrupções.
- Para cada exceção gerada durante a execução de um programa a ação é interromper a execução do programa e invocar uma rotina para o tratamento da exceção.

- **Interrupção X Exceção**
  - A diferença essencial entre exceções e interrupções é a seguinte: exceções são síncronas com o programa e interrupções são assíncronas.
  - Se o programa for executado um milhão de vezes com a mesma entrada, as exceções ocorrerão no mesmo lugar toda vez, mas as interrupções podem variar, dependendo, por exemplo, de quando, exatamente, quem estiver no terminal pressionar a tecla Enter.
  - A razão para a possibilidade de reprodução de exceções e a impossibilidade dessa reprodução é que as exceções são causadas diretamente pelo programa, e interrupções, no máximo, são causadas indiretamente pelo programa.

Outras Técnicas Para Multiprogramação

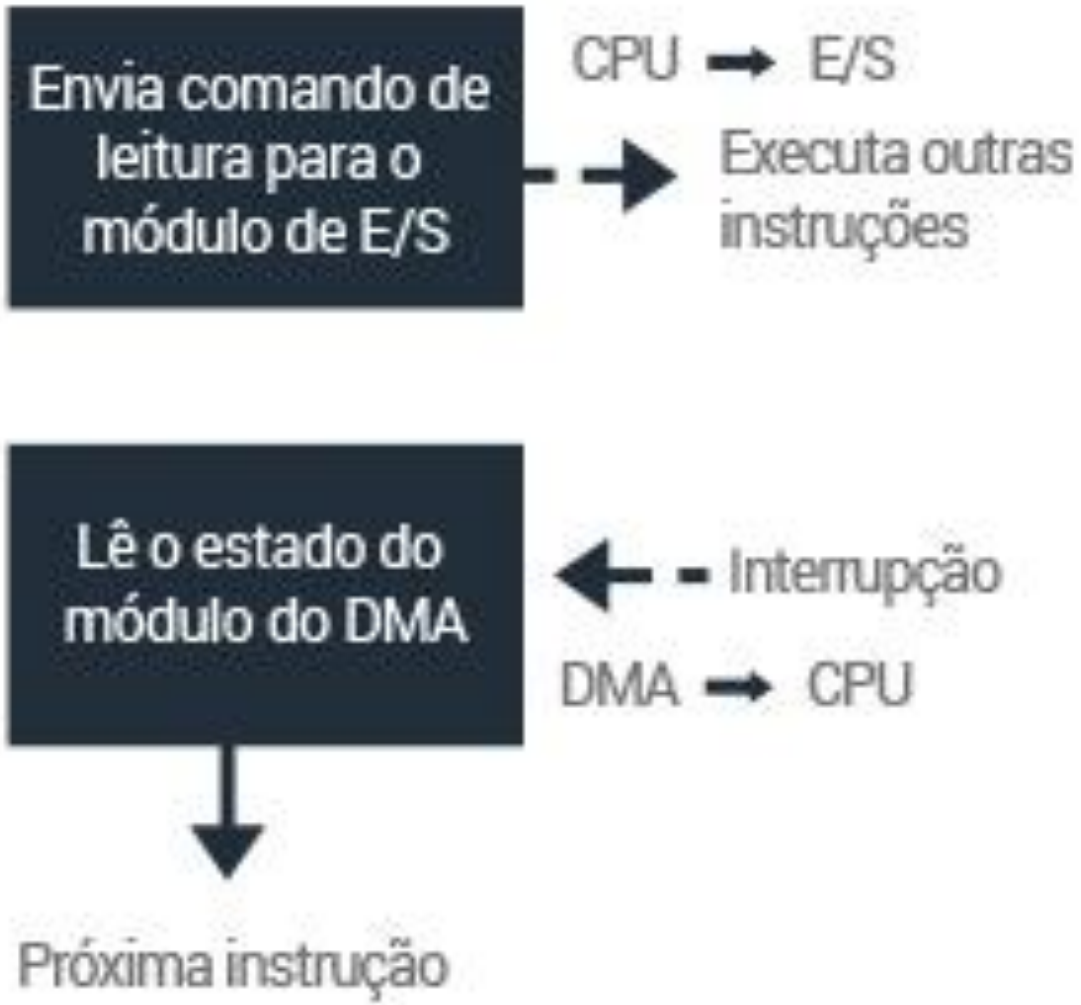
- **Operações de Entrada/Saída**
  - **Instruções de entrada/saída:**
    - Primitivo.
      - Comunicação entre a CPU e os periféricos controladas por um conjunto de instruções especiais.
      - Limitava a comunicação do processador a um conjunto particular de dispositivos.

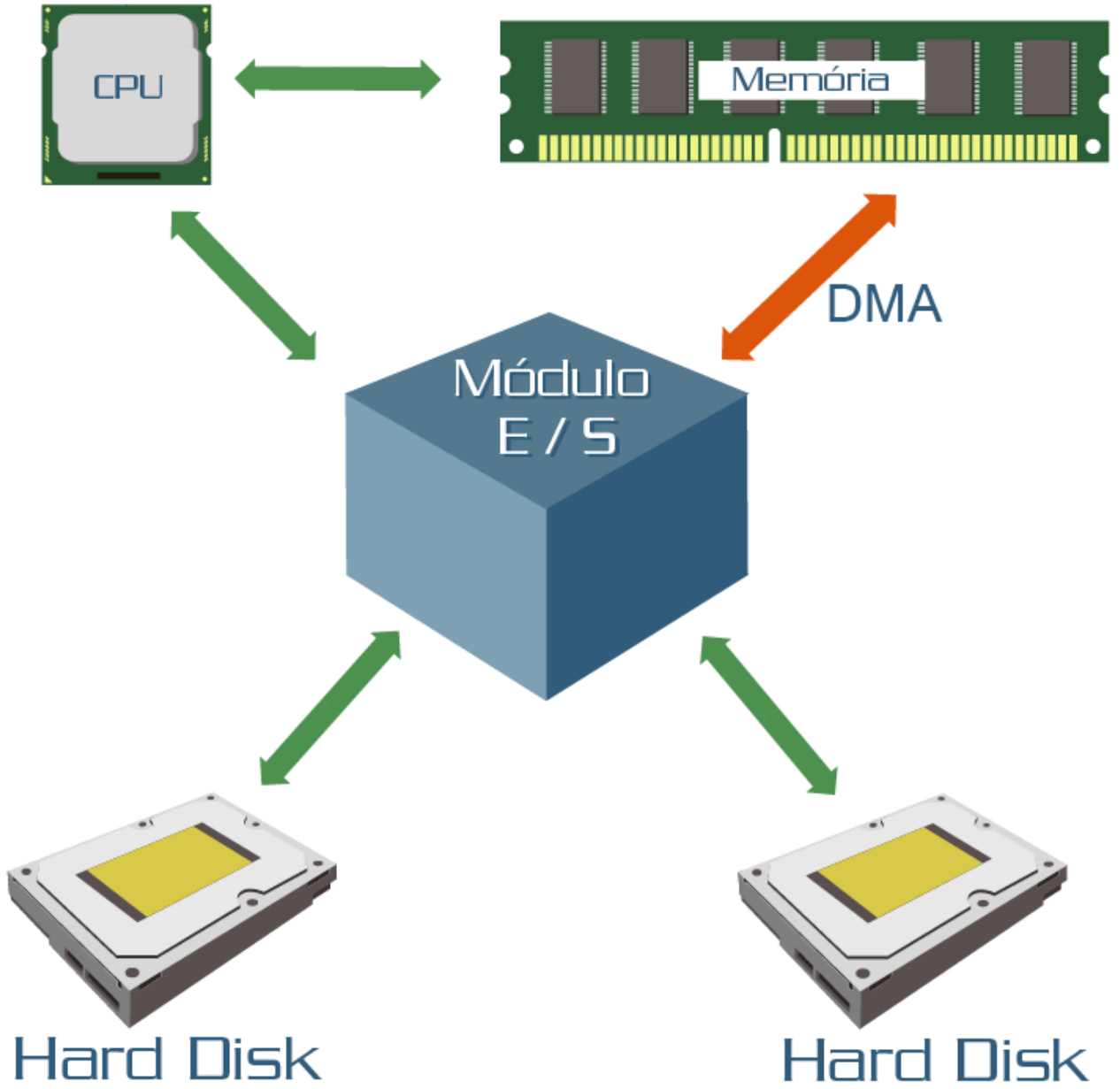
- **Controlador de interface:**
  - Mais atual
    - CPU interage independente dos dispositivos de E/S.
    - CPU não comunica diretamente com periféricos, mas através de um controlador.
    - Controle de operações de E/S pelo processador.

- **Controlada por programa:**
  - CPU sincronizada com periférico no início da operação.
  - Sistema testa periférico esperando final da operação.
  - Ocupa CPU até término da operação (busy wait).
  - Desperdício de CPU.

- **Por Polling:**
  - CPU liberada para outras tarefas.
  - Controlador testa final da operação a cada período de tempo (polling).
  - Permitiu o paralelismo e sistemas multiprogramáveis.
  - Mais eficiente.

- **DMA**
  - Polling exigiu a implementação, por parte do controlador, da Direct Memory Access (DMA).
  - Transferir blocos de dados entre memória e periféricos, sem intervenção da CPU, a não ser no início e término da operação.
  - CPU interrompida só no início e final da Operação.
  - No momento da Transferência da DMA, CPU interrompe acesso ao barramento.


Envia comando de leitura para o módulo de E/S
CPU → E/S
Executa outras instruções
Lê o estado do módulo do DMA
Interrupção
DMA → CPU
Próxima instrução


CPU
Memória
DMA
Módulo
E / S
Hard Disk
Hard Disk

Modos de Transferência DMA

- **Modo de Transferência por Bloqueio:** O controlador DMA transfere um bloco de dados de uma vez e depois libera o controle de volta ao processador.
- **Modo de Transferência por Ciclo de Pulso:** O controlador DMA transfere dados em ciclos regulares, permitindo que o processador interaja com a memória entre os ciclos.
- **Modo de Transferência por Bloqueio de Acesso:** O controlador DMA usa um mecanismo de bloqueio para garantir que a memória não seja usada pelo processador durante a transferência.

DMA – Processo de Transferência

- **Configuração:** O processador configura o controlador DMA com o endereço de memória de origem/destino e o tamanho dos dados a serem transferidos.
- **Início:** O controlador DMA é ativado para iniciar a transferência de dados.
- **Transferência:** O controlador DMA gerencia a transferência de dados diretamente entre o dispositivo e a memória, usando o barramento do sistema.
- **Finalização:** Após a transferência, o controlador DMA sinaliza ao processador que a operação foi concluída, geralmente por meio de uma interrupção.

Tipos de DMA

- **DMA de Canal Único:** Um único canal de DMA que pode gerenciar uma única transferência de cada vez.
- **DMA de Múltiplos Canais:** Vários canais de DMA para permitir a transferência simultânea de dados entre diferentes dispositivos e a memória.
- **DMA de Memória Virtual:** Utiliza endereços de memória virtual, facilitando a gestão de memória em sistemas modernos.

DMA - Exemplos Práticos

- **Transferência de Arquivos:** Em sistemas de armazenamento, como discos rígidos ou SSDs, o DMA permite transferir grandes blocos de dados rapidamente para a memória.

- **Dispositivos de Rede:** Em adaptadores de rede, o DMA pode ser usado para transferir pacotes de dados diretamente para a memória, minimizando a sobrecarga do processador.

DMA em Sistemas Modernos

- Em arquiteturas modernas, como a arquitetura de computadores com múltiplos núcleos e sistemas de cache avançados, o DMA evoluiu para se adaptar a novas necessidades e desafios, como a transferência de dados entre diferentes dispositivos e diferentes níveis de memória cache.

- O conceito de DMA é fundamental para otimizar o desempenho do sistema e melhorar a eficiência das operações de E/S em computadores modernos.

- **Processador de E/S:**
  - Evolução com memória própria.
  - Sem necessidade de programas de E/S serem carregados na Memória Principal.
  - Controle com mínima intervenção da CPU.

- **Buffering:**
  - Utilização de uma área da memória para transferência de dados entre periféricos e memória principal denomida buffer.
  - Os dado são transferidos para o buffer.
  - O dispositivo pode iniciar nova leitura enquanto a CPU manipula os dados do buffer.
  - O mesmo pode ser aplicado para operações de gravação.
  - Minimiza o problema da disparidade da velocidade de processamento e dispositivos de E/S
  - Objetiva manter CPU e dispositivos de E/S ocupados na maior parte do tempo.
  - O buffer possui uma fila **FIFO (First In First Out)** podendo conter vários registros (unidade de transferência usada no mecanismo de buffering).

- **Spooling (simultaneous pefipheral operation on-line):**
  - Surgiu no final dos anos 50.
  - Base dos Sistemas Batch
  - Antes, as Operações E/S eram lentas, deixando a CPU ociosa.
  - No spooling vários programas (jobs) eram armazenados em uma fita magnética, então eram enviados para processamento.
  - Diminuição do tempo de execução dos jobs e transição entre eles.
  - Da mesma forma um job poderia direcionar as saídas para impressora para outra fita.
  - Sistemas estritamente sequenciais devido as fitas magnéticas.
  - Mais eficiência com o surgimento de dispositivos de acesso direto, como discos e atribuição de prioridades aos jobs.

- **Reentrância:**
  - Diversos usuários podem estar executando o mesmo utilitário (compartilhado) simultaneamente.
  - Não precisa ter mais de uma cópia do mesmo utilitário na memória
  - Exige que o código reentrante não possa ser modificado por nenhum usuário enquanto está sendo executado.
  - Diversos usuários podem acessar partes diferentes do código manipulando seus próprios dados
  - Exemplo: Editores de texto, compiladores, linkers.
    - O linker é um programa que tem como objetivo principal criar um arquivo executável a partir de um ou mais arquivos objeto (módulos compilados) e outros arquivos, como bibliotecas e recursos.

- **Proteção do Sistema:**
  - Garantia de proteção de recursos compartilhados, como memória, dispositivos de E/S e CPU
  - Na memória, cada usuário deve possuir uma área de dados e códigos armazenados de forma que outros não interfiram nessas informações.
  - Numa impressão, não deve ser possível a utilização até que a impressão corrente termine.
  - O Sistema Operacional deve implementar esses mecanismos (modos de acesso).