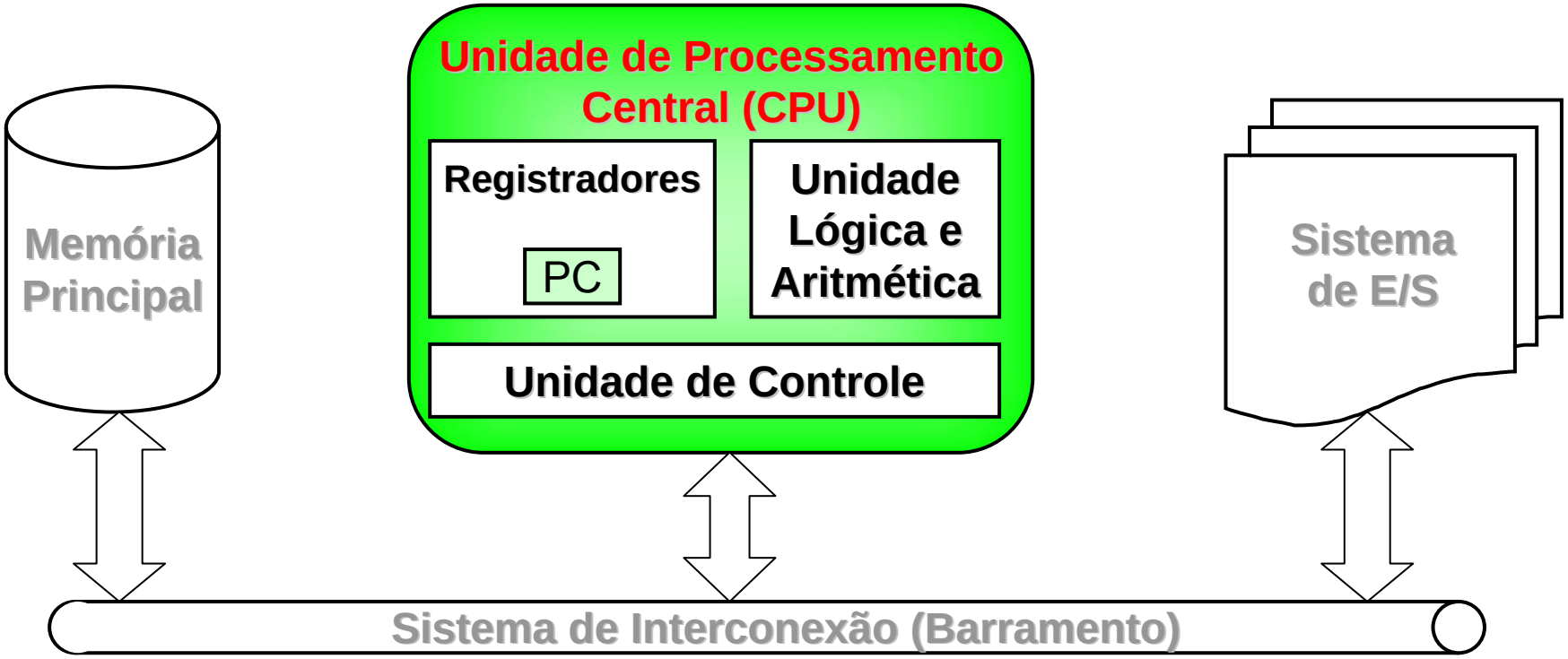
Organização de Computadores 1

3.1 – CPU: Unidade de Processamento Central

Prof. Luiz Gustavo A. Martins

Arquitetura de von Newmann

- **Unidade de Processamento Central (CPU):**



Unidade de Processamento Central (CPU)

- A CPU é o "***cérebro***" do computador.

- Funções:
  - **Interpretação e execução dos programas** da memória principal;
  - **Controle dos demais componentes**.

- Componentes:
  - Registradores
  - Unidade Lógica e Aritmética (ULA)
  - Unidade de Controle (UC)

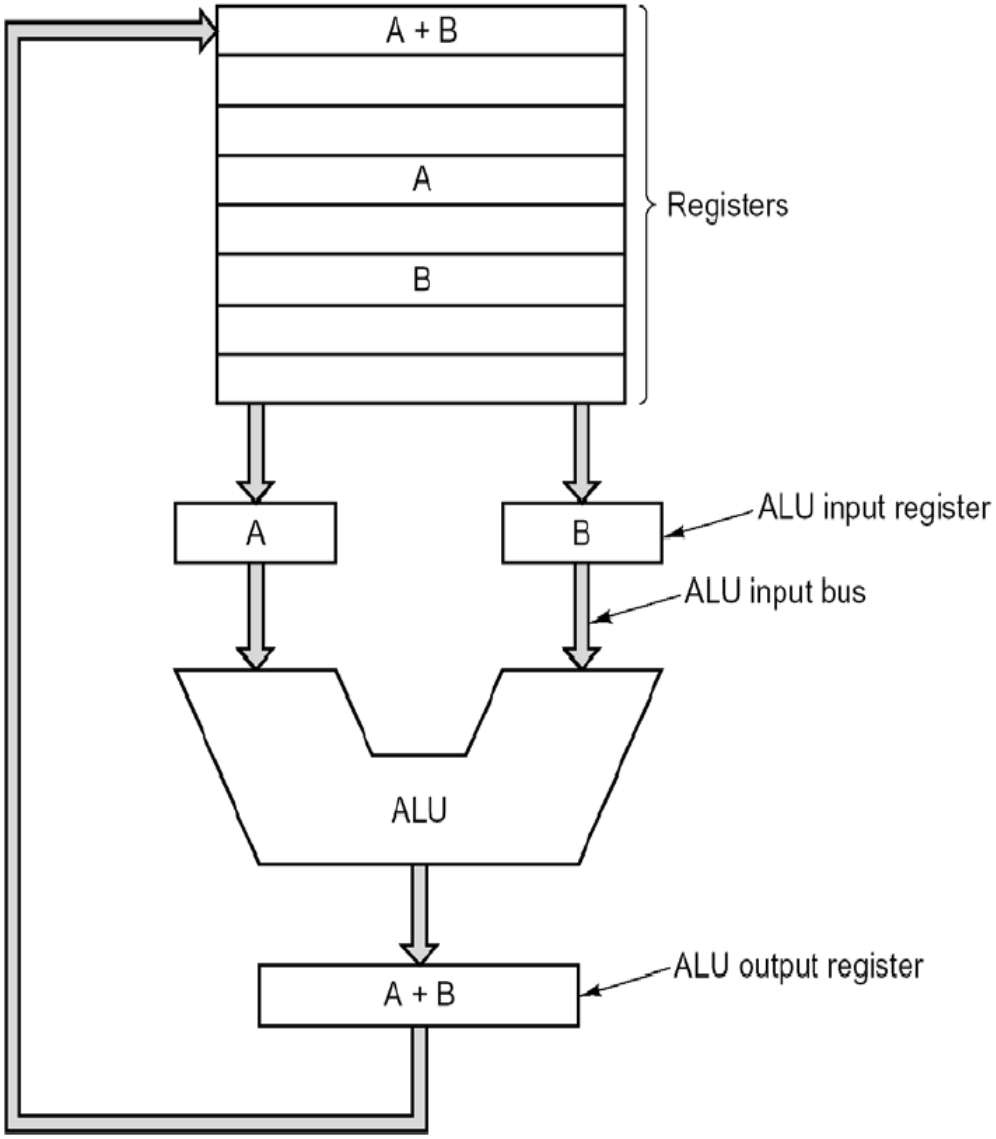
CPU: Execução de uma instrução

- **Unidade de Controle** (UC) "**dispara**" cada um das etapas de execução da instrução.
- **Registradores armazenam** temporariamente dados e instruções.
- **Unidade Lógica e Aritmética** (ULA) "**processa**" os dados e atualiza os registradores.

Unidade Lógica e Aritmética (ULA)

- A ULA é o "***núcleo***" da CPU.
- **Executa as operações** de processamento de dados.
  - Podem ser diferentes para cálculos com inteiros e ponto flutuante.
- Ativa bits especiais (***flags***) como resultado da operação.
  - **Ex:** operação nula (bit Z), operação negativa (bit N), *overflow*, etc.
- Está conectada a um grupo de registradores pelo barramento interno, formando o **caminho de dados**.
- Pode-se utilizar um **conjunto de ULAs** para a **execução paralela** de instruções.

Caminho de Dados



Registradores

- Pequenas unidades de **memória** com **alta velocidade**.
  - **Mais rápido** que as memórias principal e cache.
  - Utilizam o **barramento interno** da CPU.
- Armazenamento **temporário** de dados, instruções e endereços em utilização pelo processador.
- Possuem **diferentes funções**, mas têm **um uso bem definido** dentro da arquitetura.
- Possibilitam operações de **leitura e escrita**.
  - Alguns permitem acesso indireto no nível ISA.

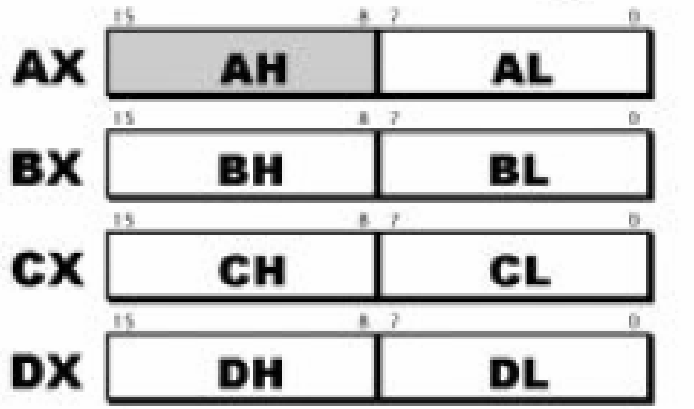
Registradores: Classificação

- Registradores de **uso geral**:
  - Utilizados para **armazenar dados** que serão **processados ou produzidos pela ULA**.
    - **Ex:** AX-DX, AC, MQ.
  - Coletivamente são chamados **conjunto de registradores de dados** (*data register file*).

- Registradores de **controle**:
  - Utilizados **no controle** das operações pela CPU e nas **trocas de informações** com a MP.
    - **Ex:** PC, IR, MAR, MBR.
  - Alguns desses são **invisíveis** aos programadores.

Registradores (Processador 8088/8086)

- CPU possui 14 registradores de **16 bits visíveis**.
- **4 registradores de uso geral:**
  - **AX (Acumulador):** armazena operandos e resultados dos cálculos aritméticos e lógicos.
  - **BX (Base):** armazena endereços indiretos.
  - **CX (Contador):** conta iterações de *loops* ou especifica o n° de caracteres de uma *string*.
  - **DX (Dados):** armazena *overflow* e endereço de E/S.
  - Podem ser usados como **registradores de 8 bits**:
    - **Ex: AH** e **AL** (byte alto e byte baixo de **AX**).



Registradores (Processador 8088/8086)

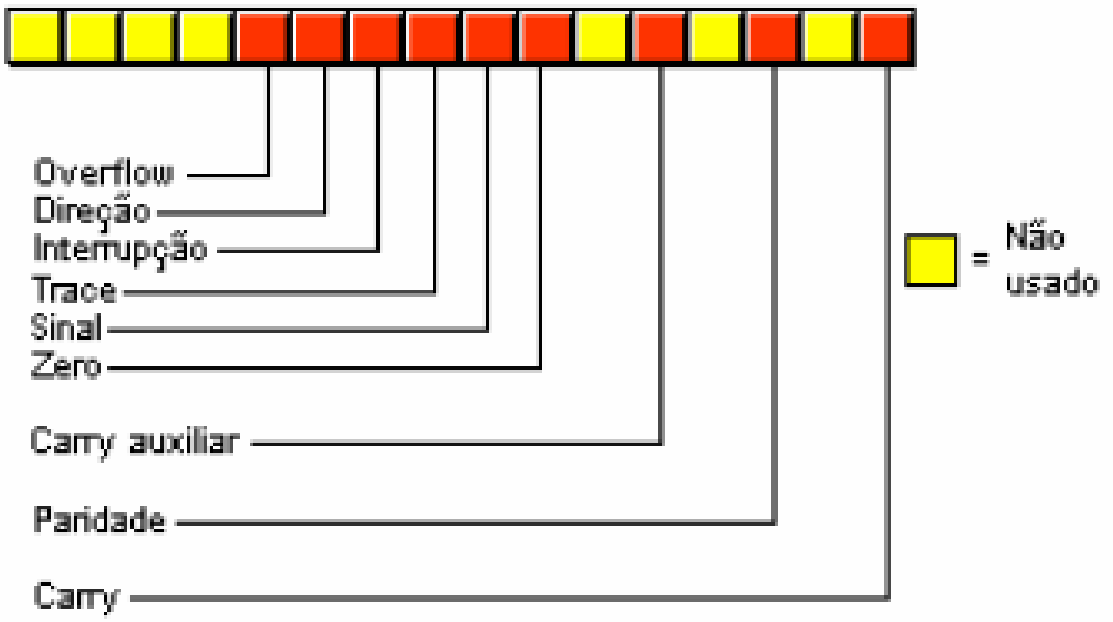
- **4 registradores de segmento:**
  - **CS (Segmento de Código):** contém o endereço da área com as **instruções** de máquina em execução.
  - **DS (Segmento de Dados):** contém o endereço da área com os **dados** do programa.
    - Geralmente aponta para as variáveis globais do programa.
  - **SS (Segmento de Pilha):** contém o endereço da área com a **pilha**. Que armazena informações importantes sobre o estado da máquina, variáveis locais, endereços de retorno e parâmetros de subrotinas.
  - **ES (Segmento Extra):** utilizado para ganhar acesso a alguma área da memória quando não é possível usar os outros registradores de segmento.
    - **Ex:** transferências de bloco de dados.

Registradores (Processador 8088/8086)

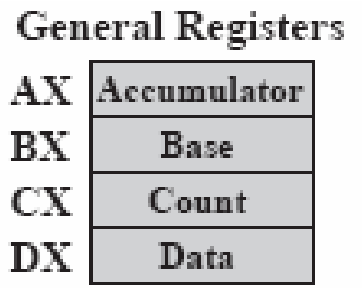
- **5 registradores de *offset*:**
  - **PC** (*Program Counter*) ou **IP** (***Instruction Pointer)*:** usado em conjunto com o **CS** para apontar a próxima instrução.
  - **SI** (***source index***) e **DI** (***destiny index)*:** utilizados para **mover blocos de bytes** de um lugar (**SI**) para outro (**DI**) e como **ponteiros para endereçamento** (junto com os registradores CS, DS, SS e ES).
  - **BP** (***Base Pointer***)**:** usado em conjunto com o **SS** para apontar a base da pilha.
    - Similar ao registrador **BX**.
    - Usado para acessar parâmetros e variáveis locais.
  - **SP** (***Stack Pointer***)**:** usado em conjunto com o **SS** para apontar o topo da pilha.

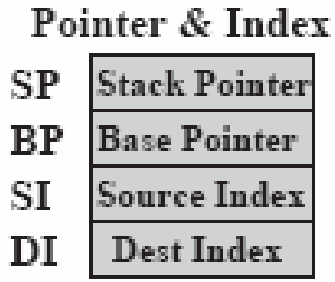
Registradores (Processador 8088/8086)

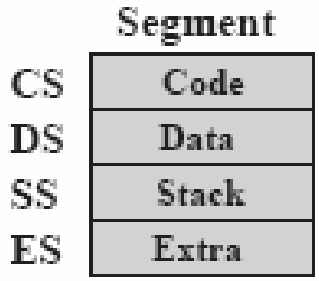
- **1 registrador de estado do processador (PSW) :**
  - Registrador especial composto por sinalizadores (***flags***) que ajudam a determinar o **estado atual** do processador.
    - Coleção de **valores de 1 bit**.
  - Apenas 9 bits são utilizados.
    - **4 mais utilizados:** ZF - zero; CF - *carry* ("vai um“) ou *borrow* (“vem um”); SF - sinal; e OF - *overflow* ou *underflow*.



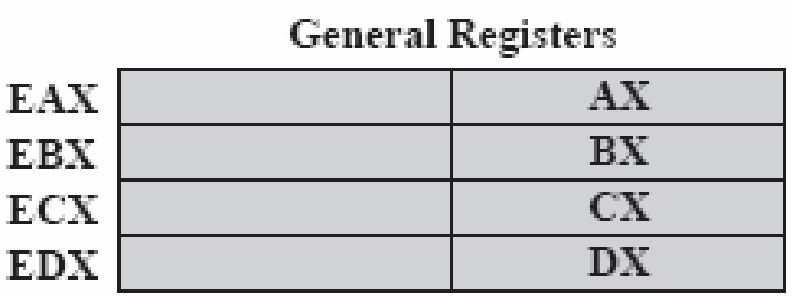
Organização dos Registradores – Família Intel

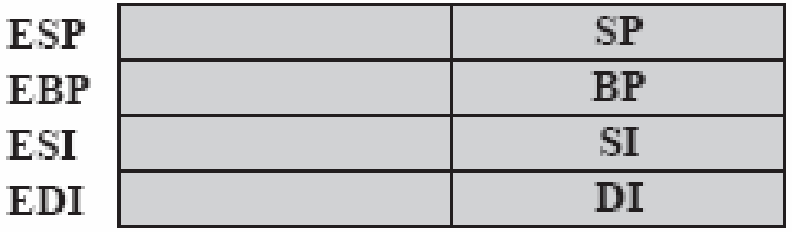






8086
(16 bits)





80386 – Pentium II
(32 bits)

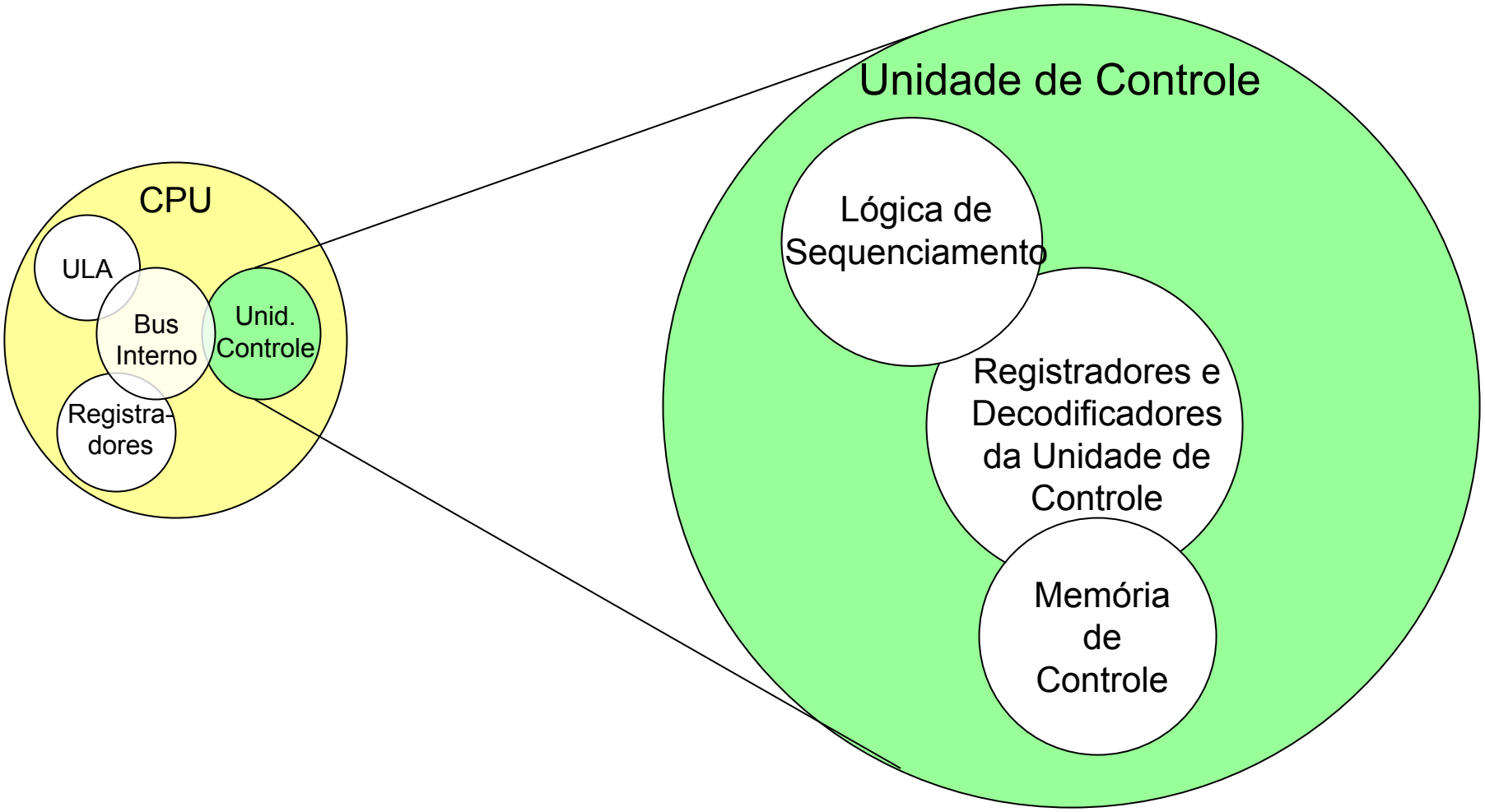
Unidade de Controle (UC)

- **Gerencia os recursos** disponíveis **e o fluxo de dados** entre os componentes.

- **Controla a execução** das instruções pela CPU:
  - **Busca** as instruções ISA na memória principal.
  - **Decodificação** das instruções (geração dos sinais de controle correspondentes).
  - **Seqüenciamento** das operações.
  - Disparo da **execução** (envio dos sinais de controle).

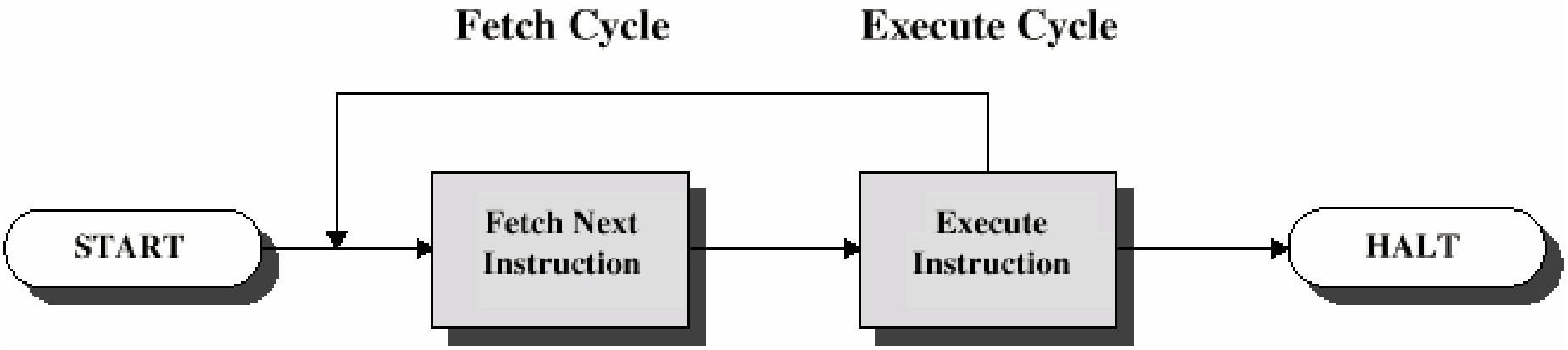
- Representa uma das partes mais **difíceis de ser projetada** em um computador, devido à complexidade dos processadores.

Visão Estrutural da Unidade de Controle



Ciclo de Instrução

- **Ciclo de Busca-Execução (*fetch-execute*):**
  1. Busca a instrução (memória → IR);
  2. Altera PC para indicar a próxima instrução;
  3. Decodifica a instrução atual;
  4. Determina o endereço e busca o operando na memória (quando necessário);
  5. Executa a operação (sinais de controle);
  6. Armazena os resultados;
  7. Repete passos anteriores.



CPU: Ciclo de Instrução

- COMPORTAMENTO_CPU()

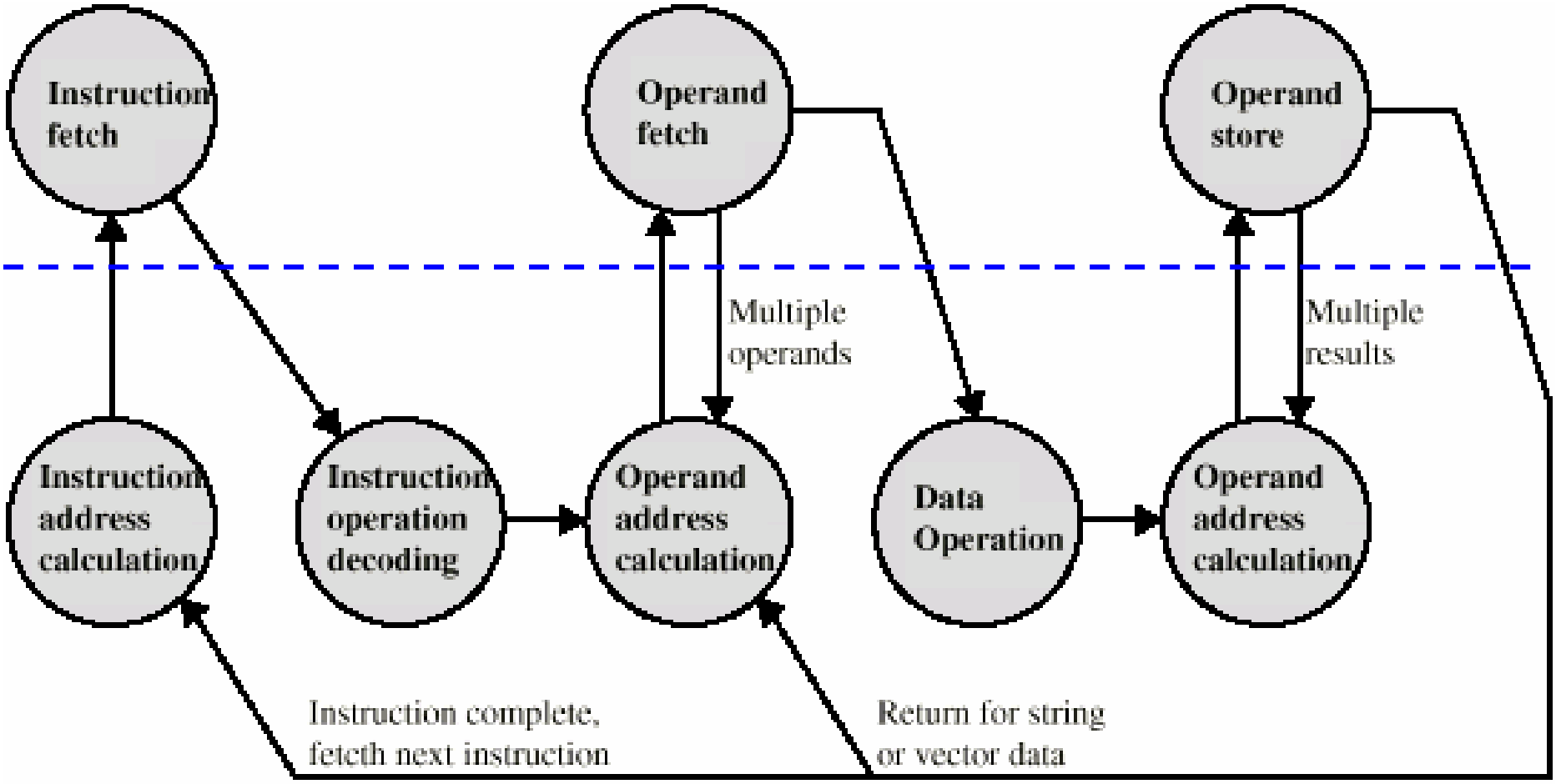
```
1.   while CPU = ativa do
2.       MAR ← PC
3.       MBR ← MEMÓRIA[MAR]
4.       IR ← MBR:OpCode
5.       PC ← PC + incremento
6.       DECODIFICA INSTRUÇÃO
7.       if IR = ADD then
8.           MAR ← MBR:Endereço
9.           MBR ← MEMÓRIA[MAR]
10.          ACC ← ACC + MBR
11.      else if IR = JUMP then
12.          PC ← MBR:Endereço
                •
                •
                •
```

Ciclo de busca da instrução (*fetch*)

Ciclo de execução da instrução (*execute*)

Ciclo de Instruções: Diagrama de Estado

Estados envolvem **transferência de valores** entre processador e memória ou E/S



Estados envolvem **operações internas** ao processador

Categorização das Operações (*Execute*)

- **Movimentação processador – memória:**
  - Transferência de dados entre CPU e memória principal.
- **Movimentação processador - E/S:**
  - Transferência de dados entre CPU e módulos de E/S.
- **Processamento de dados:**
  - Realiza alguma operação lógica ou aritmética nos dados.
- **Operações de controle:**
  - Alteração na seqüência de execução de instruções.
    - Ex: desvios condicionais e não-condicionais (*jump*)
- **Qualquer combinação das operações acima.**

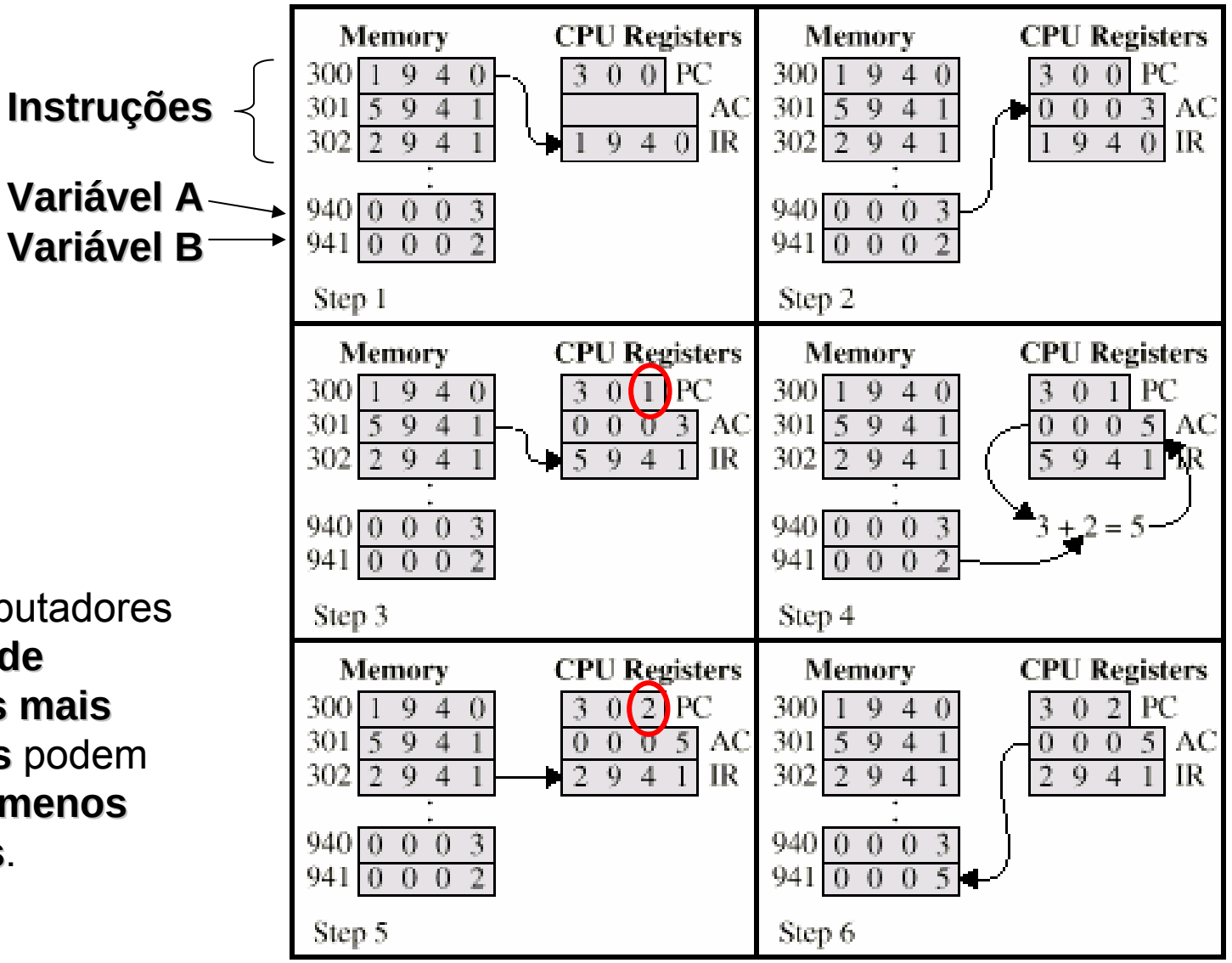
Exemplo de Execução de um Programa

- Máquina Hipotética (32 bits):
  - **Formato da instrução:**

  - **Formato de um dado inteiro:**

  - **Registradores internos da CPU:**
    - PC (*Program Counter*): endereçamento da próx. instrução
    - IR (*Instruction Register*): instrução a ser executada
    - AC (*Accumulator*): armazenamento temporário
  - **Lista parcial de *OpCode*:**
    - 0001 = transferência memória → AC (*load*)
    - 0010 = transferência AC → memória (*store*)
    - 0011 = transferência E/S → AC (*load*)
    - 0101 = adição do conteúdo da memória em AC
    - 0111 = transferência AC → E/S (*store*)

Exemplo de Execução de um Programa

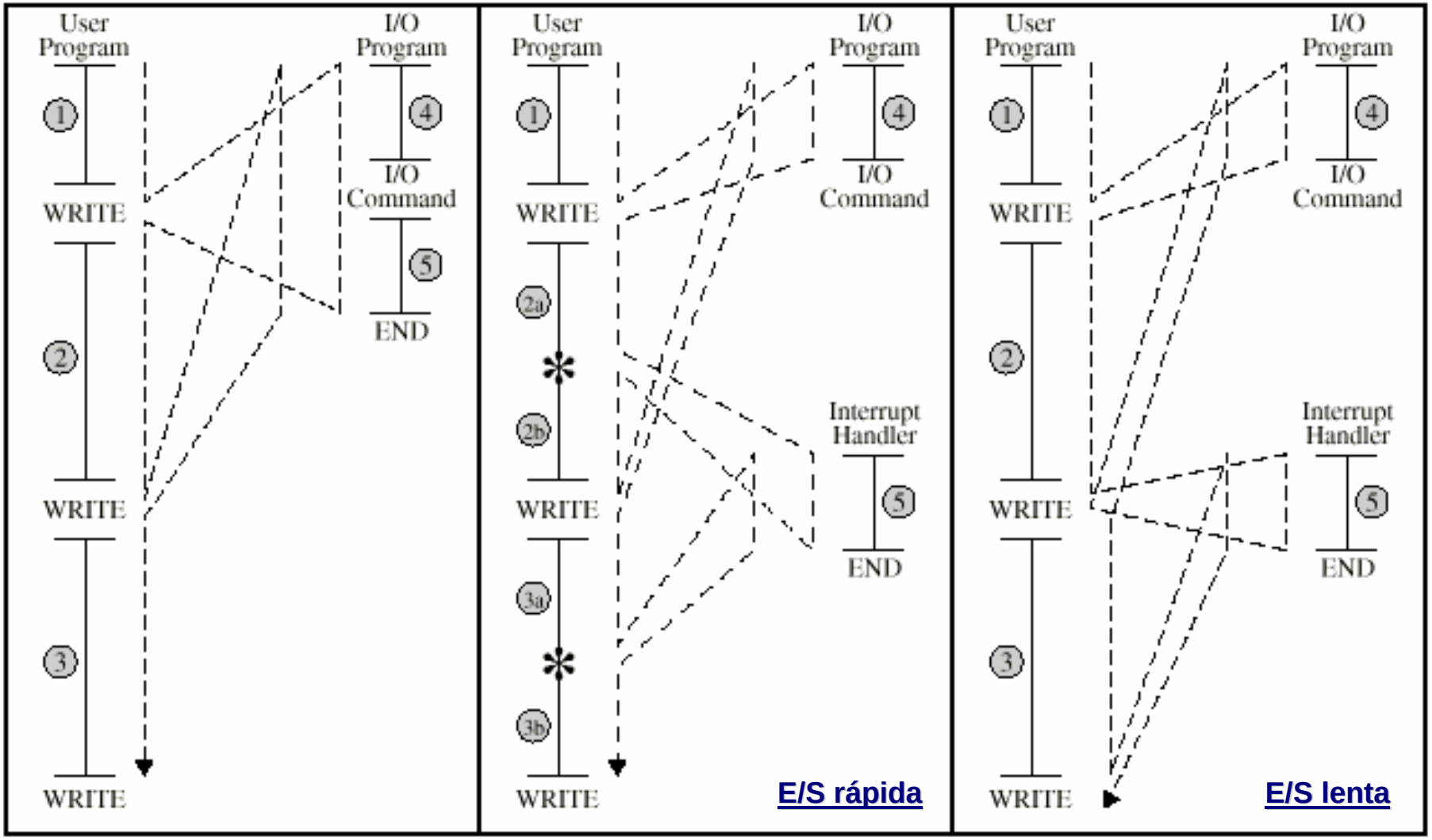


OBS: computadores com **conj. de instruções mais complexas** podem demandar **menos operações**.

Interrupção

- Mecanismos pelos quais outros componentes podem **interromper a seqüência normal** do processamento.
- Visa melhorar a eficiência do processamento.
- **Fontes de interrupções mais comuns:**
  - **Programa (Software):** gerada por alguma condição que ocorra como resultado da execução de uma instrução.
    - Ex: *overflow*, divisão por zero, instrução ilegal etc.
  - **Timer:** gerada pelo processamento interno do relógio (*timer*).
    - Usado em **sistemas multi-tarefa *preemptivos*** para executar certas funções a **intervalos regulares de tempo**.
  - **E/S:** gerada pelo módulo de E/S para sinalizar a conclusão de uma operação ou a ocorrência de uma situação de erro.
  - **Falha de Hardware:** gerada na ocorrência de uma falha.
    - Ex: queda de energia, erro de paridade de memória

Controle do Fluxo do Programa

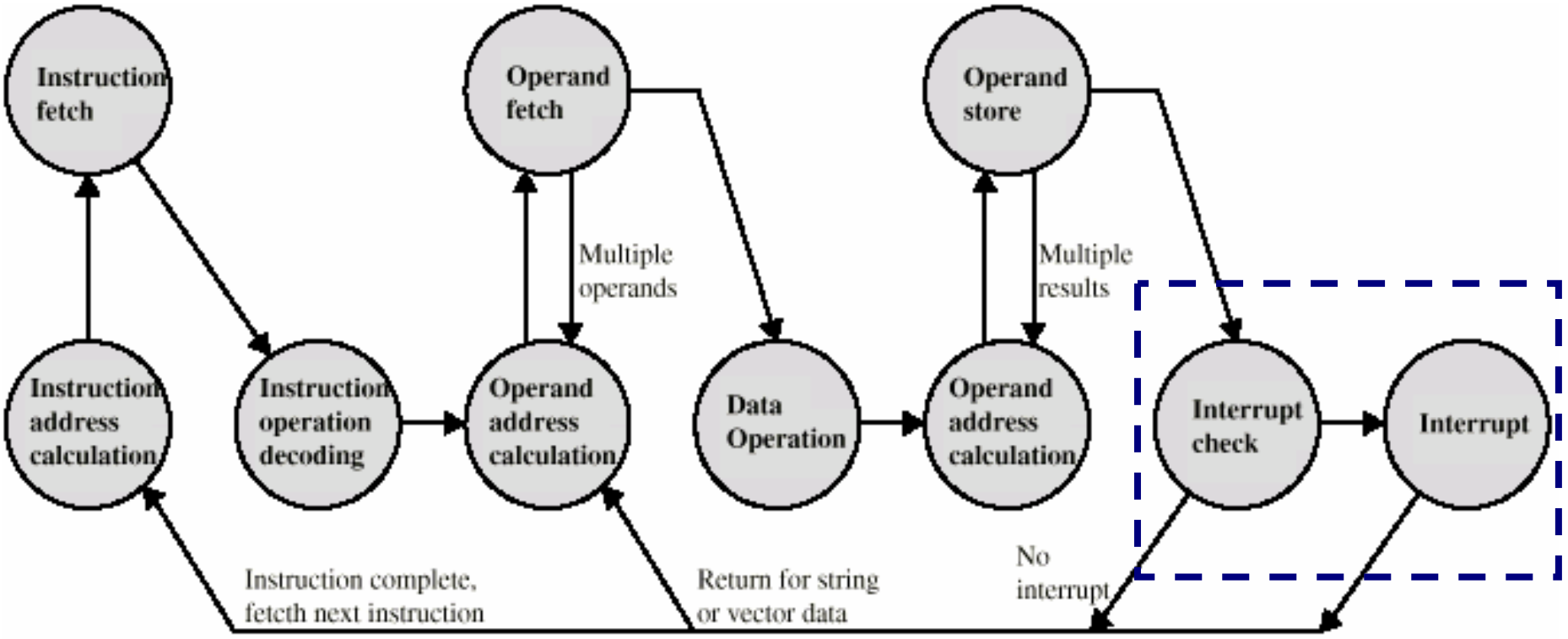


(a) Sem interrupção (b) Interrupção: espera curta (c) Interrupção: espera longa

Ciclo de Interrupção

- Acrescentado ao **ciclo de instrução**.
- Processador verifica se há interrupção:
  - Indicado por um **sinal de interrupção**.
- Se não há interrupção, busca a próxima instrução na memória.
- Se houver interrupção pendente:
  - Suspende a execução do programa corrente.
  - Salva o contexto na **pilha**.
  - Configura ***PC*** com o endereço de início da **rotina de tratamento da interrupção** (*interrupt handler routine*).
  - Processa a interrupção.
  - Restaura o contexto.
  - Continua a execução do programa interrompido.

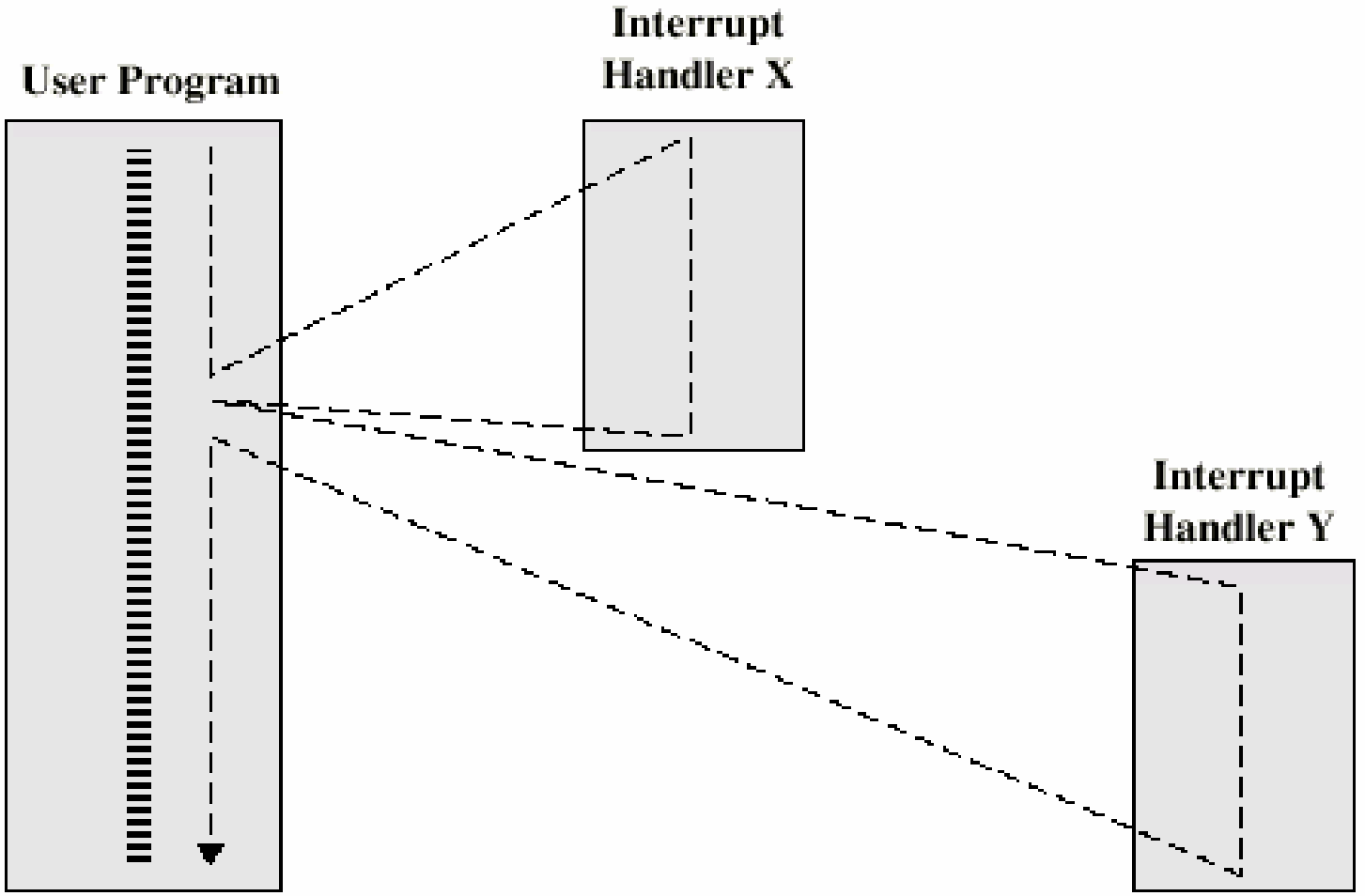
Ciclo de Instruções com Interrupção



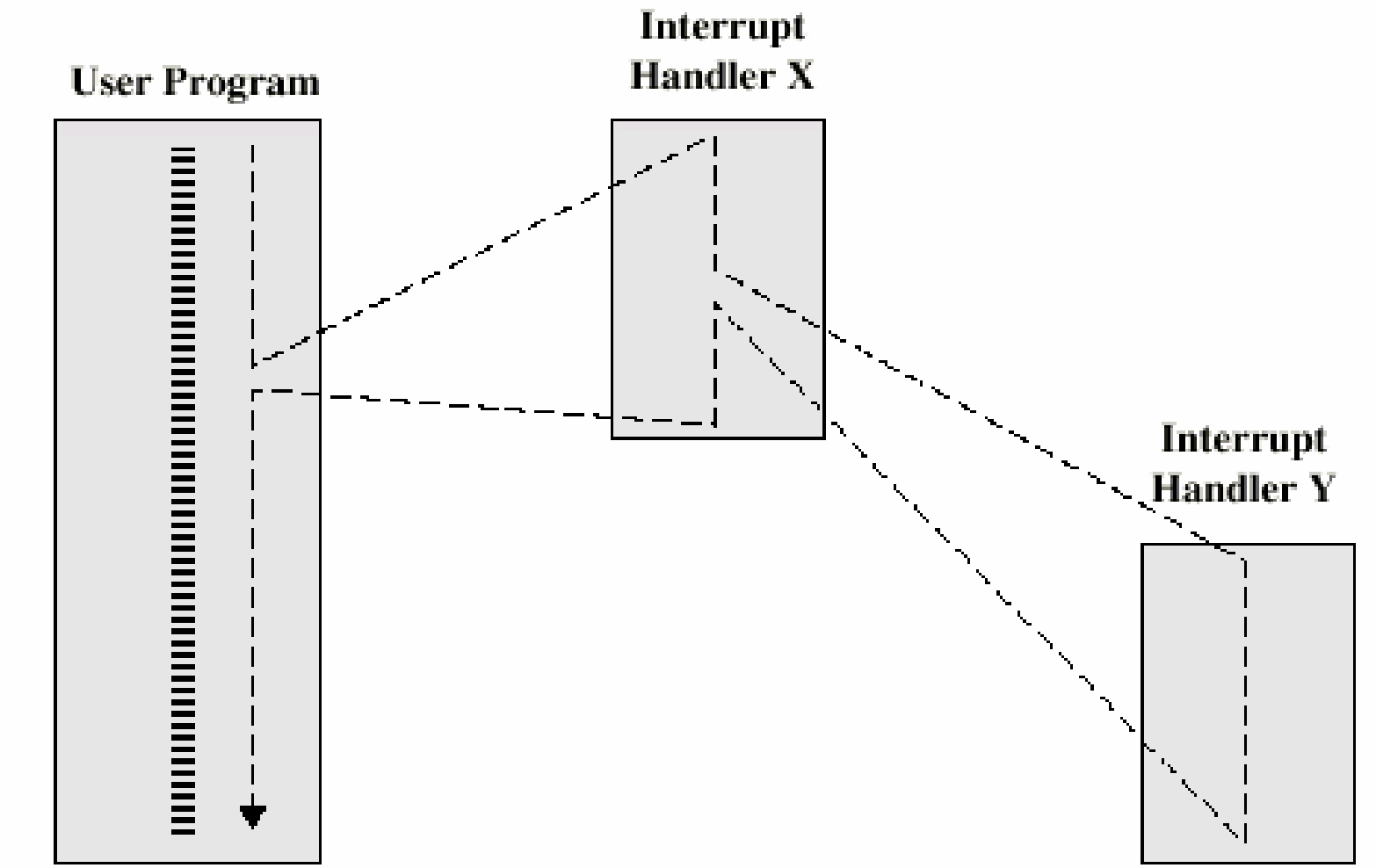
Múltiplas Interrupções

- **Desabilitar interrupções:**
  - O processador **ignora futuras interrupções** enquanto processa uma interrupção.
  - Interrupção são **manipuladas na seqüência** que elas acontecem.
- **Definir prioridades:**
  - Interrupções de baixa prioridade são interrompidas por interrupções de alta prioridade.
  - Quando a interrupção de mais alta prioridade foi processada, o processador retorna a interrupção anterior.

Múltiplas Interrupções: execução seqüencial



Múltiplas Interrupções: execução aninhada



Sinal de *Clock* (ciclo)

- Utilizado para **atender as relações de tempo requeridas** nas operações (**sincronismo**).
- Novas operações básicas são iniciadas em um **novo ciclo de *clock***.
- A execução de uma instrução consome um certo nº de ciclos de *clock*.
  - Varia de acordo com o **nº de operações básicas requeridas e o tempo de execução de cada** uma delas.
- O tamanho do ciclo de *clock* é um dos fatores que determinam o **desempenho de um processador**.
  - < tamanho do ciclo ⇒ < tempo de execução ⇒ > nº de instruções/seg.