UNIDADE II

Organização de Computadores

Prof. Dr. Alexandre Bozolan

Unidade central de processamento

- O processador é considerado o “cérebro” do computador, controlando suas tarefas, como processar, gravar ou interpretar dados/instruções, operando sobre números binários (0 e 1).

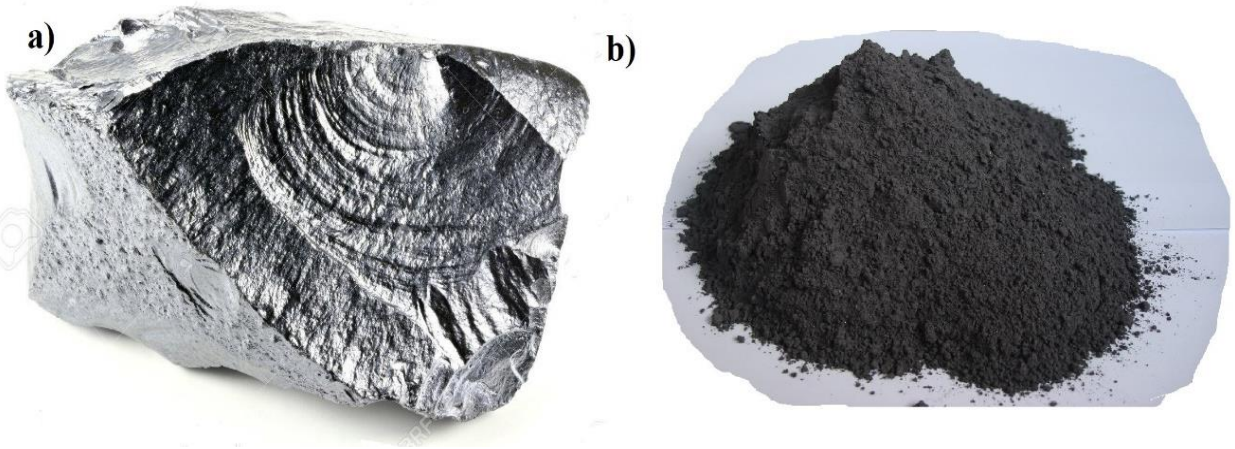
- As CPUs são empacotadas em um *chip*, constituído por um núcleo de silício, ligado a um conjunto de pinos.

Fonte: kitguru.net

Processo de fabricação da CPU

- O processo para a fabricação de um processador envolve vários estágios.

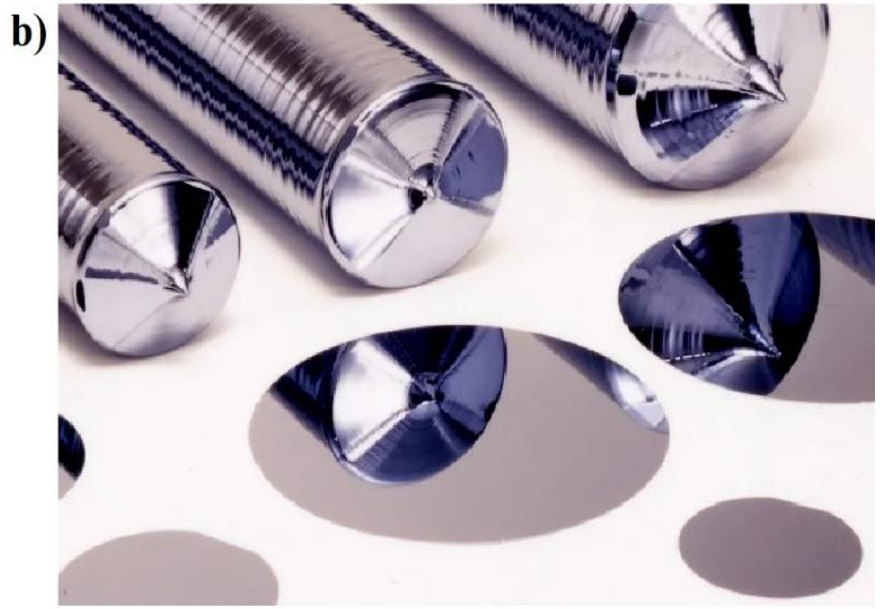
- Desde a obtenção do silício em um alto grau de pureza até o estágio final de empacotamento do processador.



Fontes: dreamstime.com
edgetech.com

Processo de fabricação da CPU

- Após a obtenção do silício em pó, ele é inserido em uma máquina de crescimento epitaxial para que seja aquecido a aproximadamente 1000 ºC e, através de um processo de rotação, obtém-se a pré-forma.

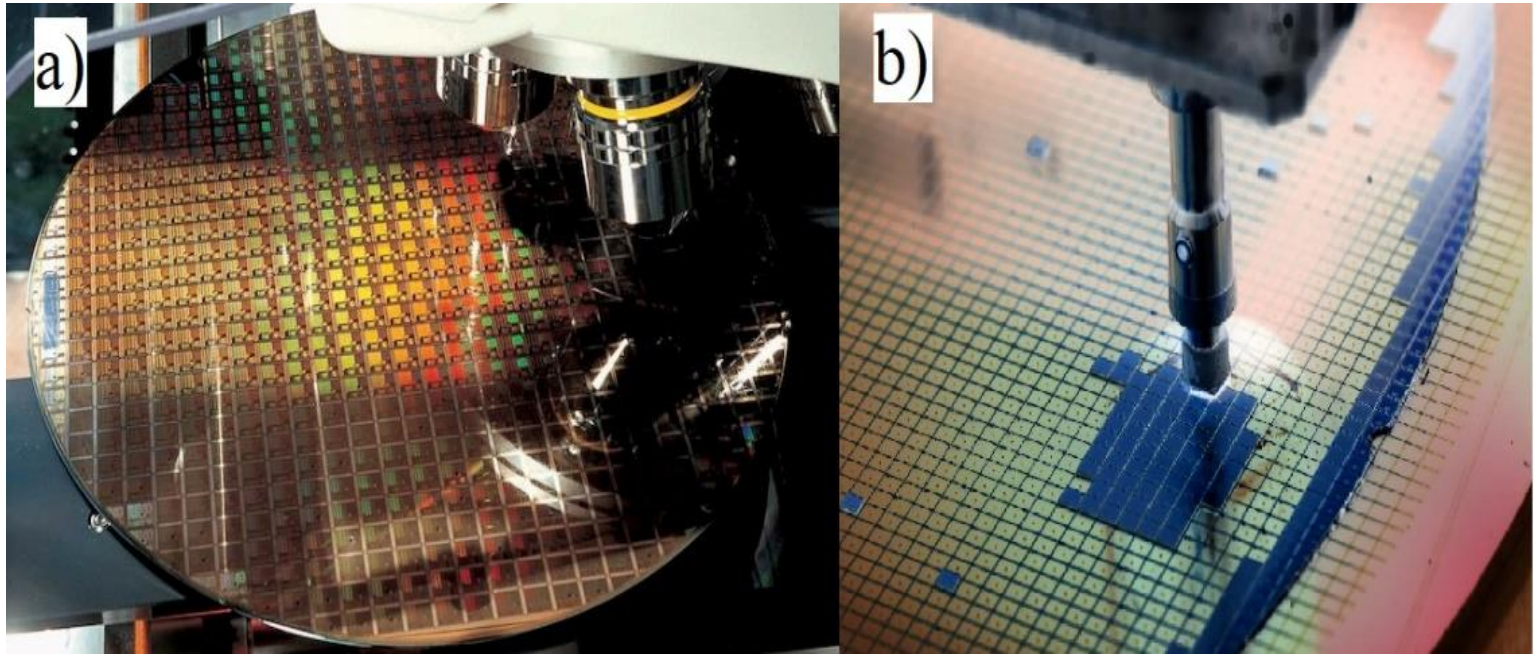


Fonte: chipsetc.com

Processo de fabricação da CPU

- Essa pré-forma é então fatiada transversalmente, facilitando o próximo estágio da fabricação, que será a litografia (impressão de circuitos eletrônicos no *wafer*).

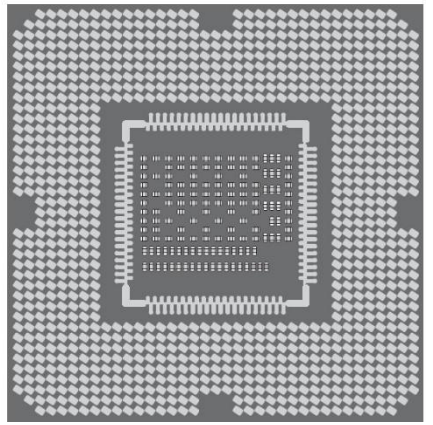
- Na sequência, é possível observar o próximo estágio de fabricação do processador, em que se retira um pequeno pedaço do *wafer*, que será empacotado num *chip*.



Fonte: chipsetc.com

Chips de CPU

- Um *chip* de CPU é basicamente dividido em três categorias: endereços, dados e controle.
- Eles são conectados a pinos diretamente no barramento da placa-mãe.

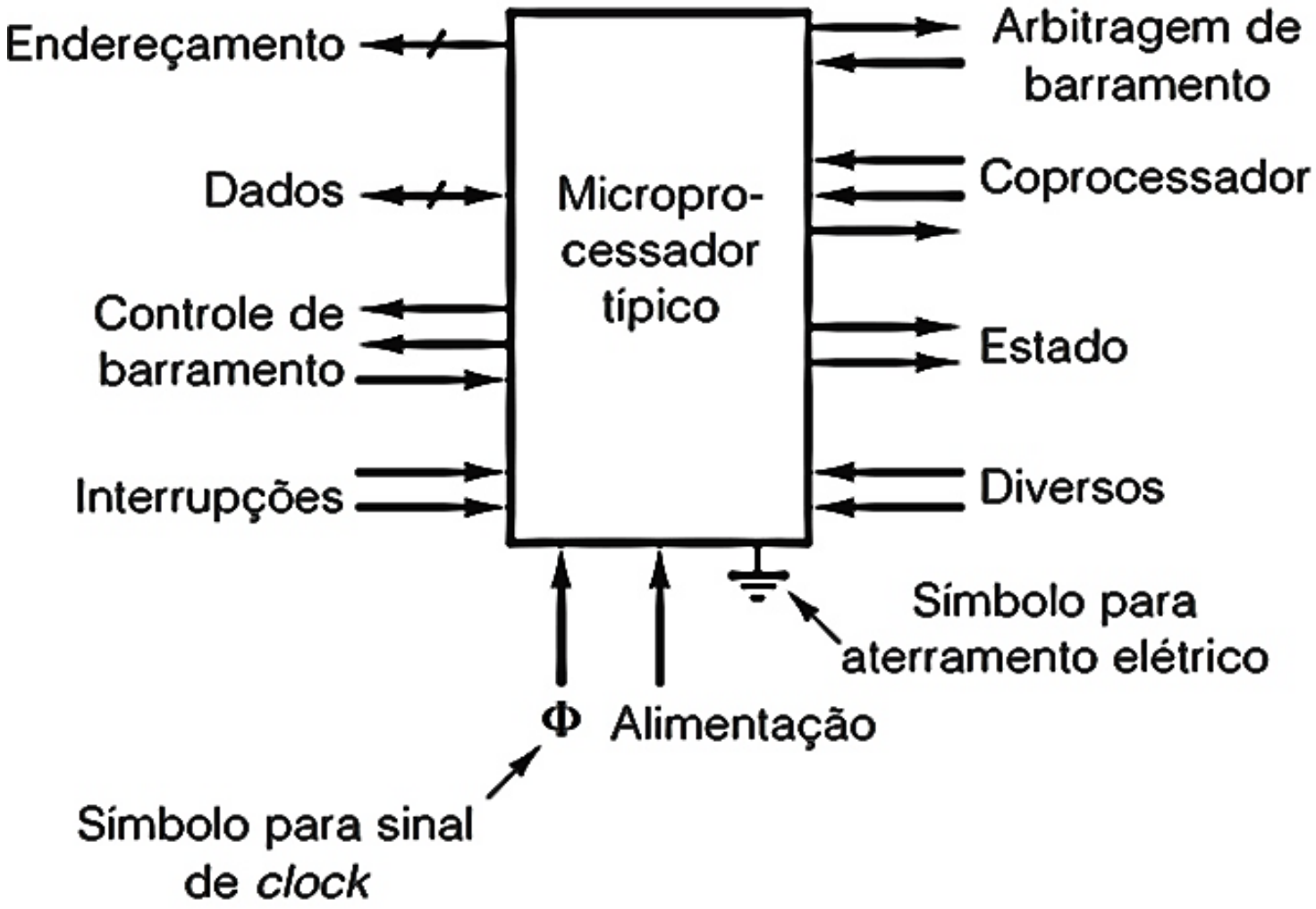


Fonte: chipsetc.com

- Ao buscar uma instrução na memória, a CPU coloca o endereço da posição de memória desejada daquela instrução em seus pinos de endereços.
  - Daí ela ativa uma ou mais linhas de controle com a finalidade de informar a memória que ela necessita ler uma palavra.
  - Assim, a memória responde colocando a palavra requisitada nos pinos de dados da CPU, ativando uma *flag* informando o que acabou de ser realizado.

Chips de CPU

- Pinagem lógica de uma CPU.



Fonte: TANEMBAUM e AUSTIN (2013, p. 147).

Unidade lógica e aritmética

- A ULA é o principal dispositivo do processador e realiza todos os cálculos binários do computador.

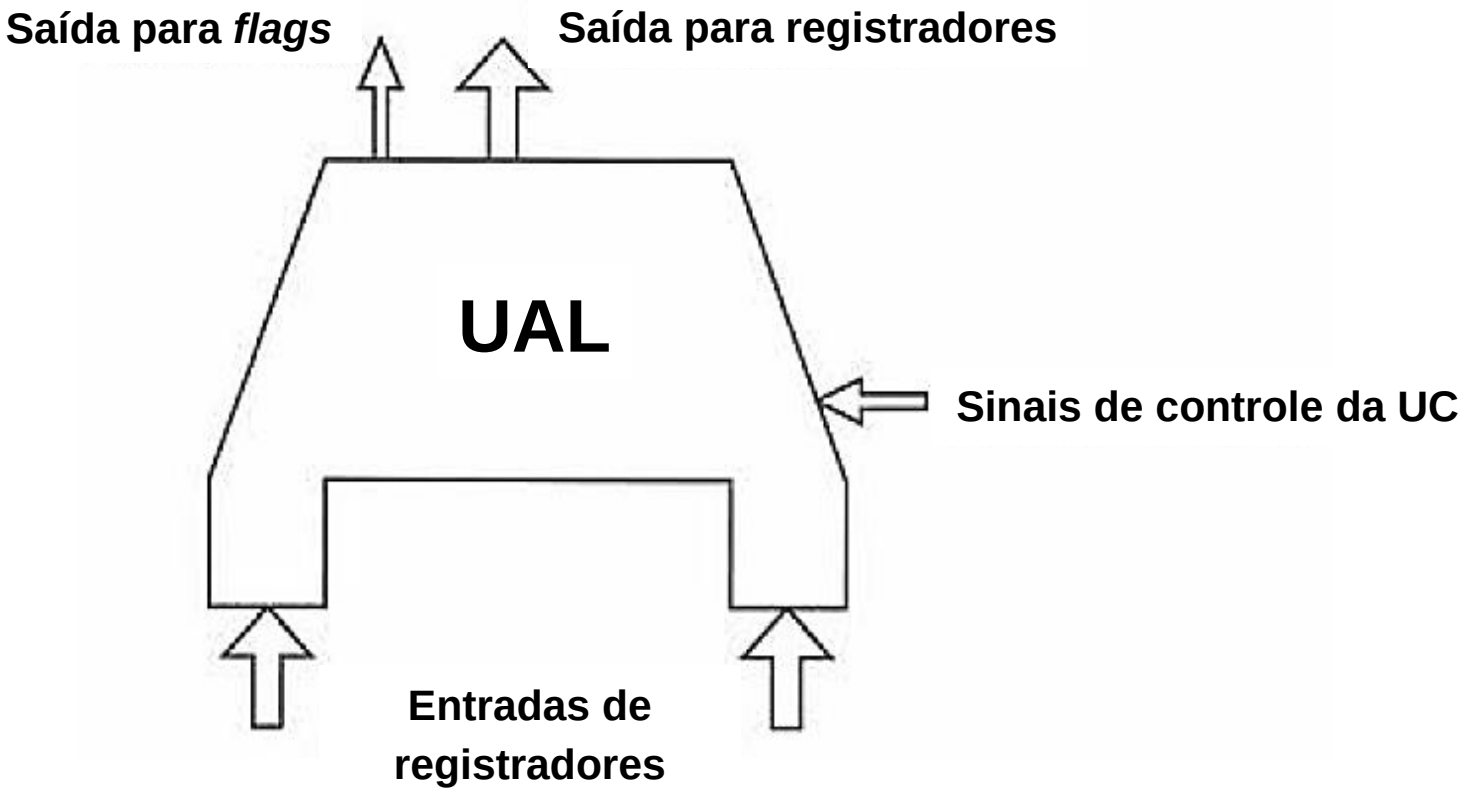
- Ela é formada por um conjunto de circuitos lógicos e componentes eletrônicos que, ao se integrarem, realizam operações, incluindo operações lógicas, operações aritméticas, operações de deslocamento, incremento e complemento.

- Uma ULA geralmente possui duas entradas de dados conectadas à saída, entrada para sinais de controle e para determinação da operação a ser realizada.

    - Além disso possui saídas de comunicação com os registradores e para sinalização de *flags*.

Unidade lógica e aritmética

- Alguns fabricantes de processadores atuais como a Intel têm substituído o nome ULA por Unidade de Cálculo ou Unidade de Execução, outros como a AMD têm chamado a ULA de UI – unidade integrada (IU – *Integer Unit*).



Fonte: MONTEIRO (2019, p. 164).

Unidade de controle

- A UC é considerada o dispositivo mais complexo da CPU.

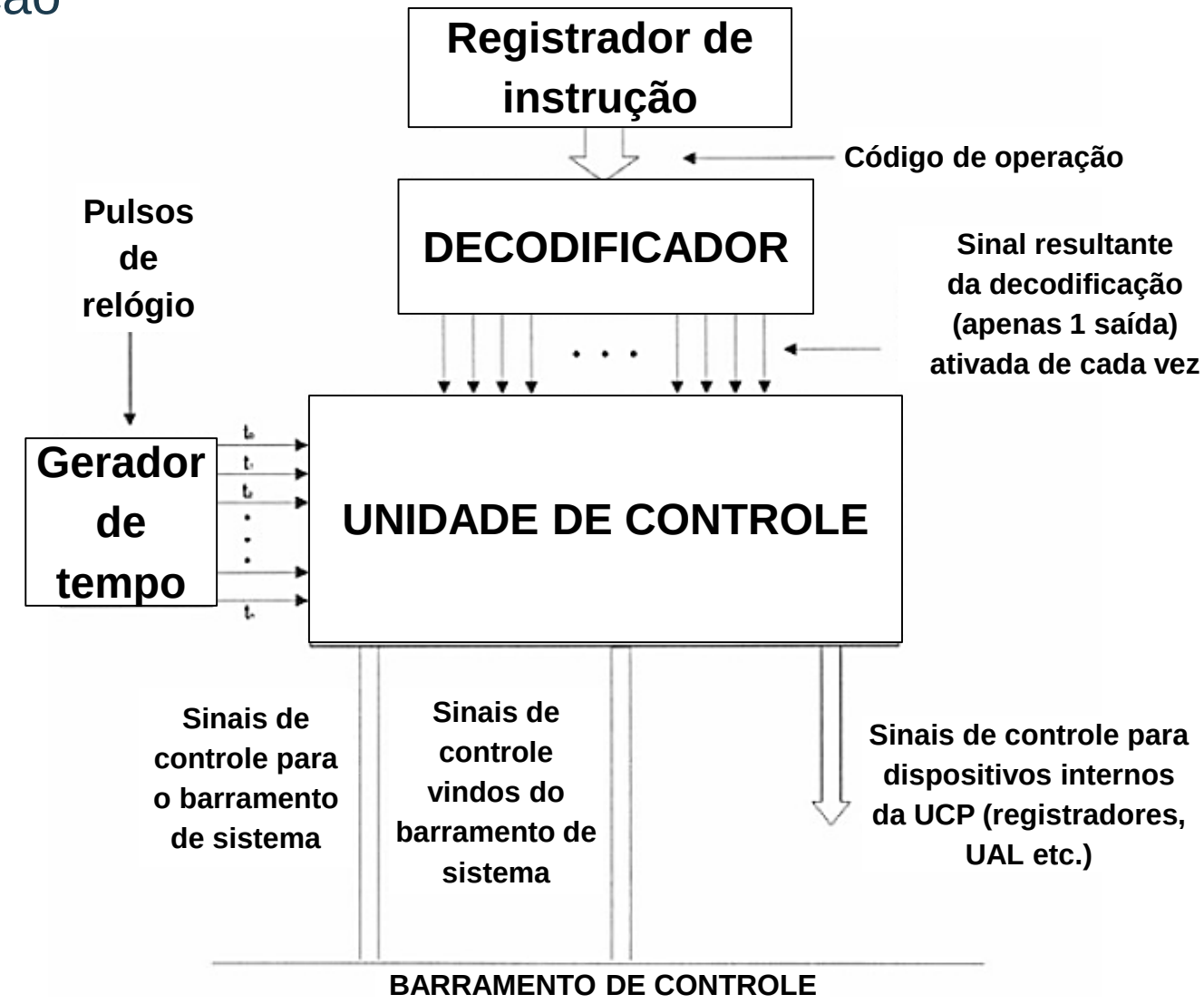
- É responsável pela movimentação de dados e instruções no processador através de sinais de controle sincronizados pelo *clock* (contador de tempo que regula e sincroniza a geração de pulsos na CPU).

- A unidade de controle opera sobre micro-operações (pequenos passos no ciclo de instruções) em que a cada início de um ciclo ocorrerá uma busca (*fetch*) da instrução solicitada.

    - Cada micro-operação é inicializada por um pulso, oriundo da UC, em decorrência de uma programação prévia, realizada diretamente no *hardware*.

Unidade de controle

- Diagrama de bloco da função de controle.



Fonte: MONTEIRO (2019, p. 172).

Interatividade

O processo de fabricação de um processador envolve vários estágios, desde a obtenção do silício em pó até o estágio final de empacotamento do processador no *chip*. Dentre os estágios da fabricação, qual é o responsável pela “impressão” dos circuitos eletrônicos no *wafer*?

a) Rotação.
b) Aquecimento.
c) Crescimento epitaxial.
d) Empacotamento.
e) Litografia.

Resposta

O processo de fabricação de um processador envolve vários estágios, desde a obtenção do silício em pó até o estágio final de empacotamento do processador no *chip*. Dentre os estágios da fabricação, qual é o responsável pela “impressão” dos circuitos eletrônicos no *wafer*?

a) Rotação.
b) Aquecimento.
c) Crescimento epitaxial.
d) Empacotamento.
e) Litografia.

Arquitetura von Neumann

- No ano de 1946, o matemático húngaro John von Neumann trabalhava no Instituto de Estudos Avançados da Universidade de Princeton, nos EUA, realizando modificações no projeto do computador ENIAC.

- Ele verificou que as tarefas de processamento e armazenamento de dados no ENIAC não eram muito eficazes.

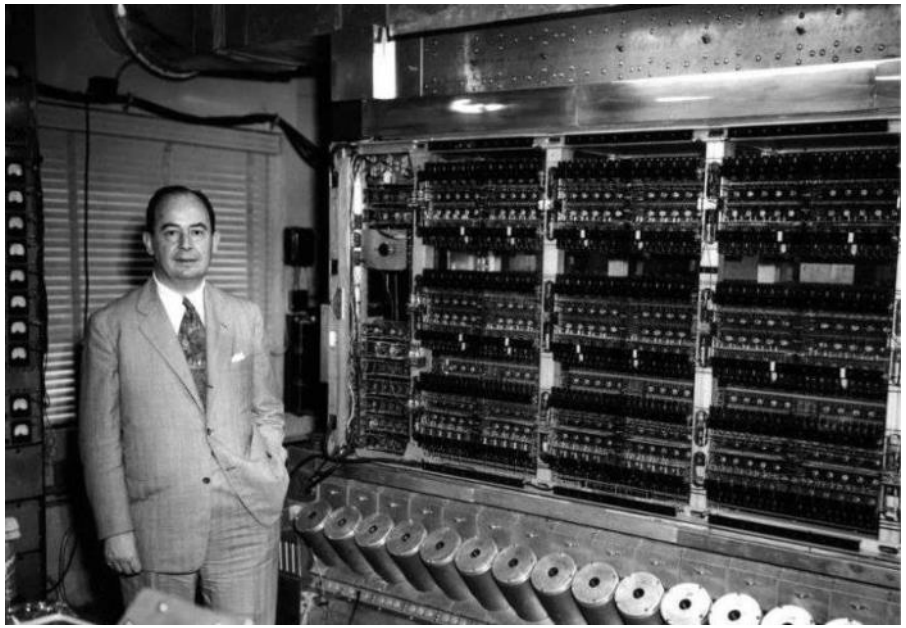
- Von Neumann então modificou a operação do ENIAC a fim de facilitar suas operações de processamento e armazenamento de dados na memória.

Arquitetura von Neumann

- Ele introduziu o conceito de que um computador poderia obter suas instruções lendo-as diretamente da memória.

- Assim, um programa poderia ser criado ou alterado através de endereços da memória, de forma que essa ideia ficou conhecida como programa armazenado.

- A máquina de von Neumann ficou pronta em 1952 e é conhecida como o protótipo para todos os computadores modernos de uso geral.



Fonte: impa.br

Arquitetura von Neumann

- Von Neumann definiu os padrões que todos os computadores modernos devem possuir:
- Memória principal para armazenamento dos dados.
- Unidade lógica e aritmética (ULA) para realizar operações em dados binários.
- Unidade de controle para interpretar/executar todas as instruções da memória principal.
    - Dispositivos de entrada e saída (E/S), controlados pela unidade de controle.

Organização dos registradores

- Os registradores são dispositivos de memória, constituídos basicamente de portas lógicas, que fornecem armazenamento temporário para dados/instruções dentro do processador.

- Os registradores estão no topo da hierarquia da memória, pois possuem maior velocidade de transferência de dados/instruções para a CPU.

- Apesar disso, os registradores possuem uma capacidade de armazenamento menor, se comparada a outras memórias.

    - Também possuem custo de fabricação mais elevado em relação a outras memórias internas.

Organização dos registradores

Os registradores são classificados de acordo com sua funcionalidade: registradores de propósito geral e registradores de controle e estado. Os registradores de controle e estado são organizados em:

- Registrador de *buffer* de memória (MBR – *Memory Buffer Register*): é o registrador que recebe uma ou várias palavras que serão armazenadas na memória ou enviadas para alguma unidade de E/S.
    - Registrador de instrução (IR – *Instruction Register*): contém um código de operação de 8 *bits*, referente à instrução que será executada.
    - Registrador de *buffer* de instrução (IBR – *Instruction Buffer Register*): é utilizado para armazenar temporariamente a próxima instrução.

Organização dos registradores

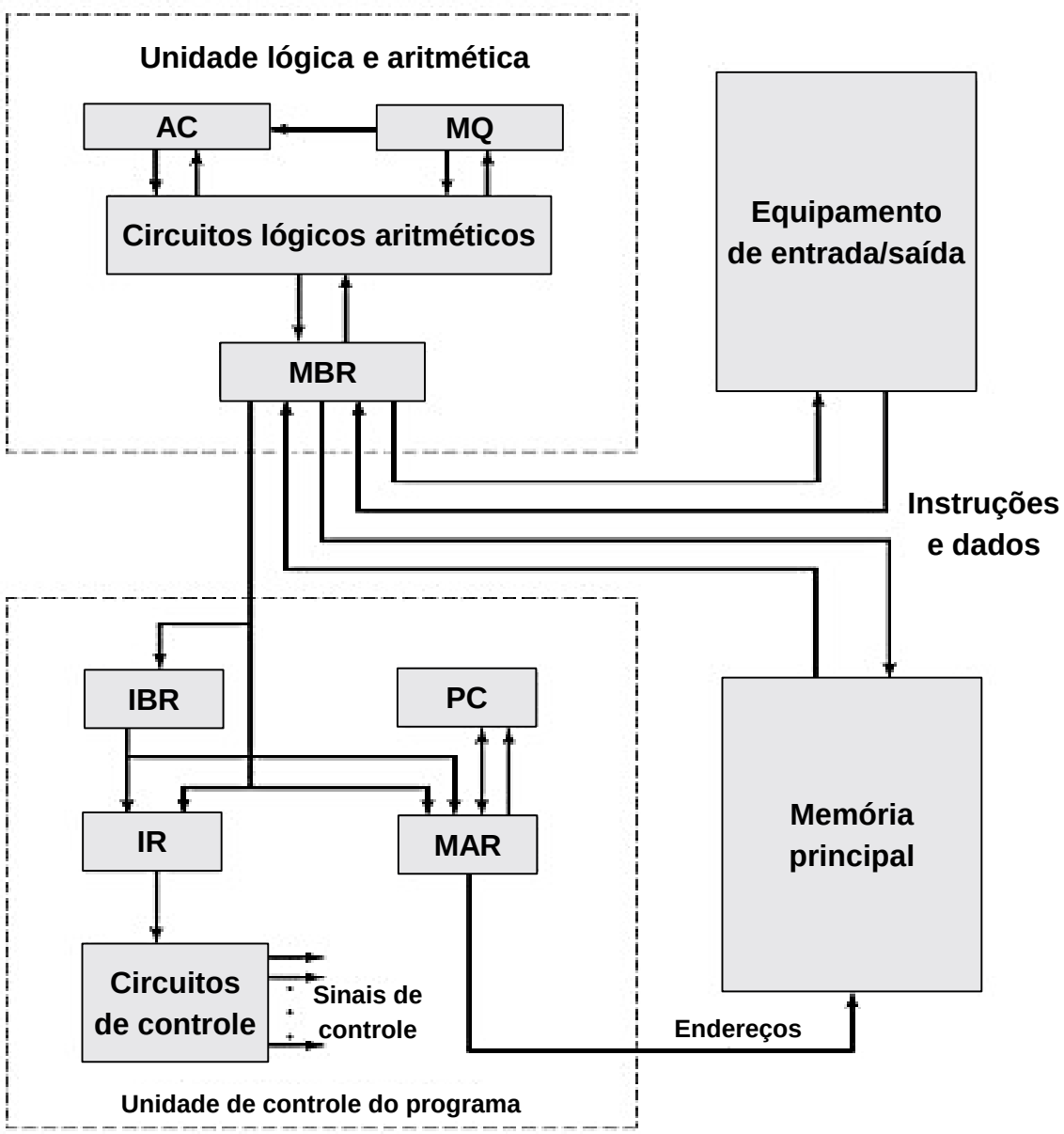
- Registrador de endereço de memória (MAR – *Memory Address Register*): é o registrador que especifica o endereço na memória principal a ser lido/escrito.

- Registrador contador de programa (PC – *Program Counter*): é o registrador que contém o endereço para busca de um par de instruções contidas na memória principal.

- Além dos registradores de controle e estado, a máquina de von Neumann também possui os registradores acumulador (AC) e o multiplicador (MQ), empregados para manter temporariamente os resultados de operações da ULA.

Organização dos registradores

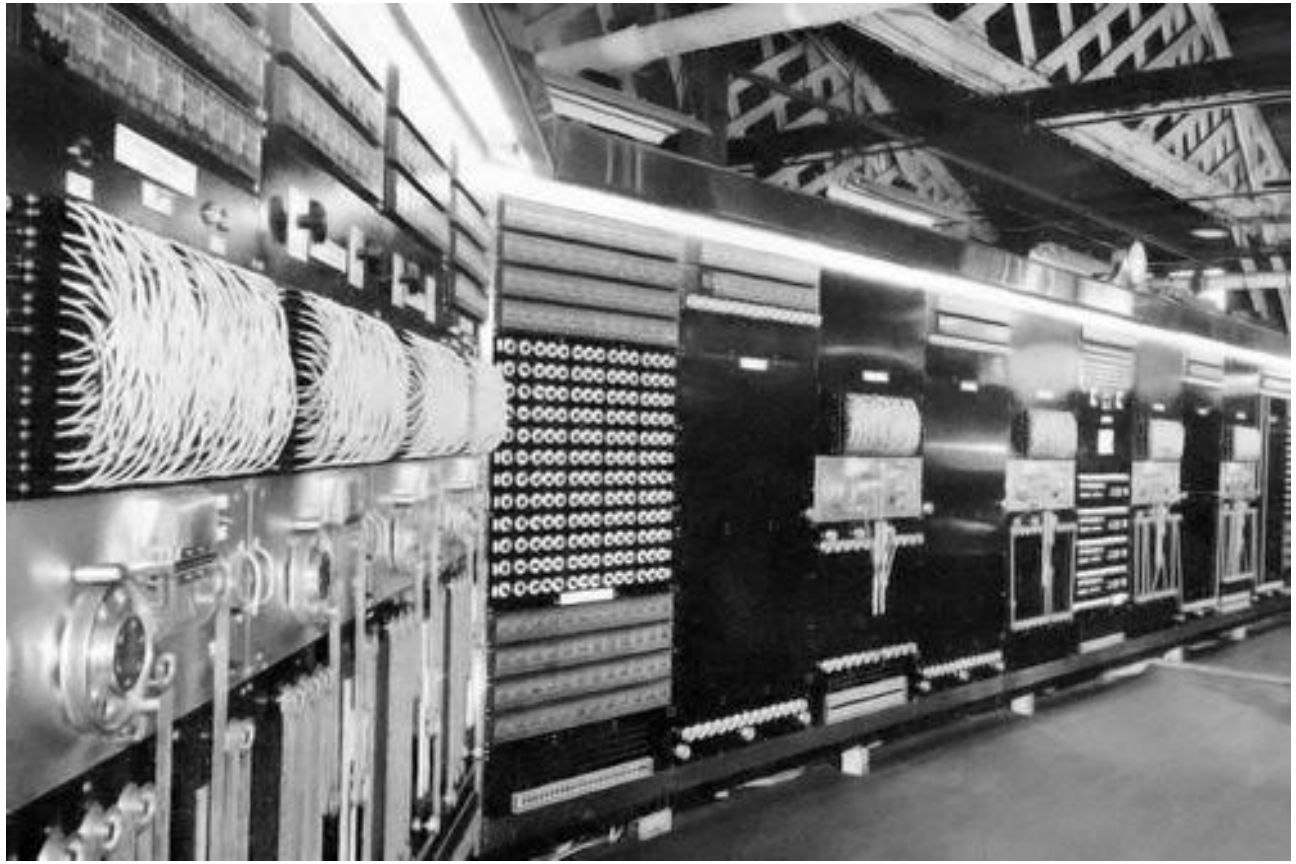
- Arquitetura von Neumann e seus registradores.



Fonte: STALLINGS (2010, p. 16).

Arquitetura Harvard

- A arquitetura Harvard teve seu início com o desenvolvimento do computador eletrônico Mark III, desenvolvido na Universidade de Harvard, em 1950.



Fonte: theymadethat.com

Arquitetura Harvard

- O Mark III possuía memórias e barramentos diferentes para dados e instruções.

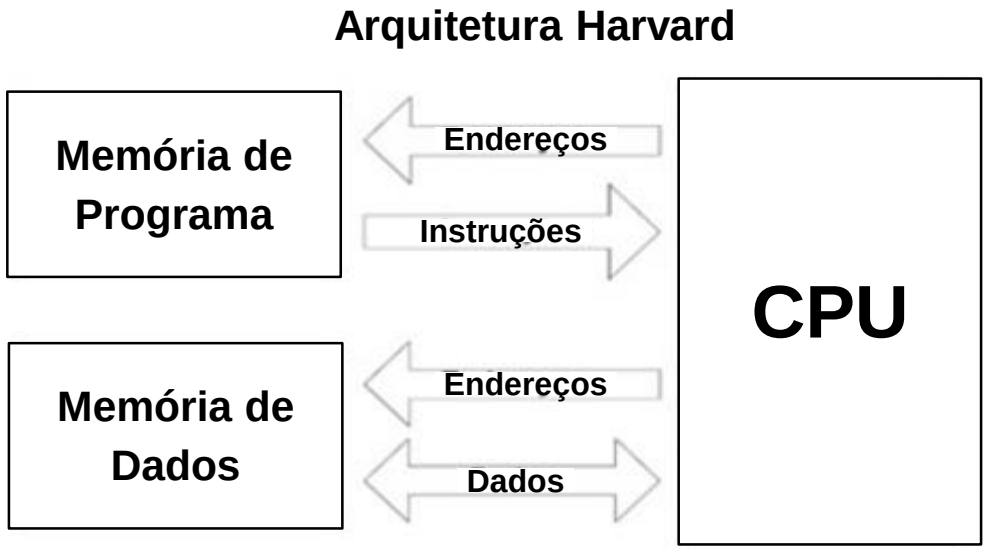
- E justamente essa é a grande diferença entre a arquitetura de von Neumann e a arquitetura Harvard, pois enquanto a arquitetura von Neumann utiliza o mesmo barramento para envio/recebimento de dados e instruções, a arquitetura Harvard utiliza barramentos diferentes.

- O uso do mesmo barramento para dados e instruções pode ocasionar gargalos na execução de tarefas de busca.

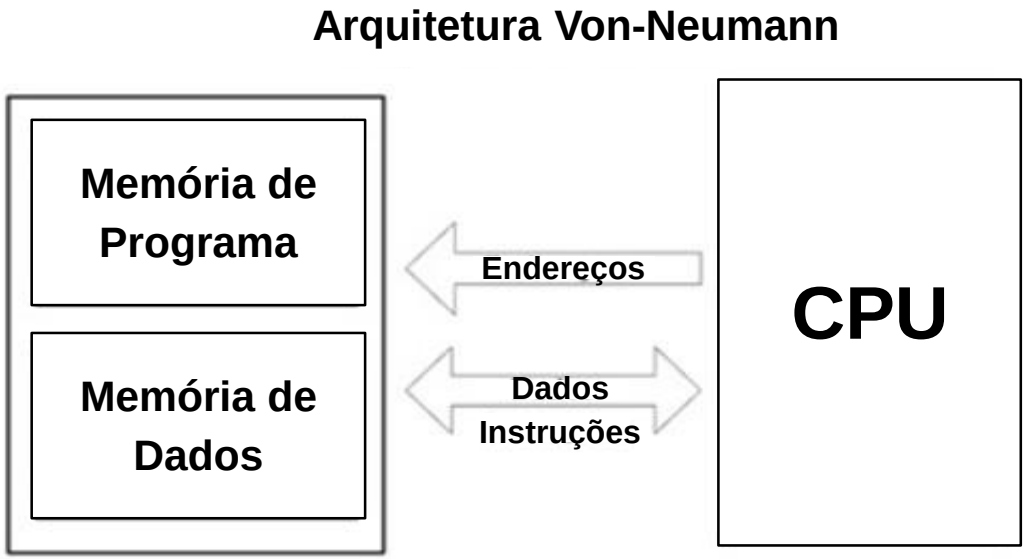
    - Como a arquitetura Harvard é capaz tanto de ler instruções ou dados ao mesmo tempo, há um grande benefício na utilização de processamento paralelo, o que aumenta a velocidade de execução das aplicações.

Arquitetura Harvard

- Comparativo entre arquitetura Harvard e arquitetura von Neumann.



Fonte: thecrazyprogrammer.com



Interatividade

A arquitetura Harvard originou-se com o desenvolvimento do computador Mark III, em 1950, e é considerada uma arquitetura concorrente à von Neumann. Dentre as alternativas a seguir, assinale qual delas aponta a principal diferença entre as duas arquiteturas.

a) A arquitetura Harvard possui memória cache interna e externa.

b) A arquitetura Harvard utiliza barramentos diferentes para dados e instruções.

c) A arquitetura Harvard utiliza barramentos em série ao invés de paralelo.

d) A arquitetura Harvard possui memória RAM e memória ROM.

e) A arquitetura Harvard utiliza BIOS diferentes para *input* e *output*.

Resposta

A arquitetura Harvard originou-se com o desenvolvimento do computador Mark III, em 1950, e é considerada uma arquitetura concorrente à von Neumann. Dentre as alternativas a seguir, assinale qual delas aponta a principal diferença entre as duas arquiteturas.

a) A arquitetura Harvard possui memória cache interna e externa.

b) A arquitetura Harvard utiliza barramentos diferentes para dados e instruções.

c) A arquitetura Harvard utiliza barramentos em série ao invés de paralelo.

d) A arquitetura Harvard possui memória RAM e memória ROM.

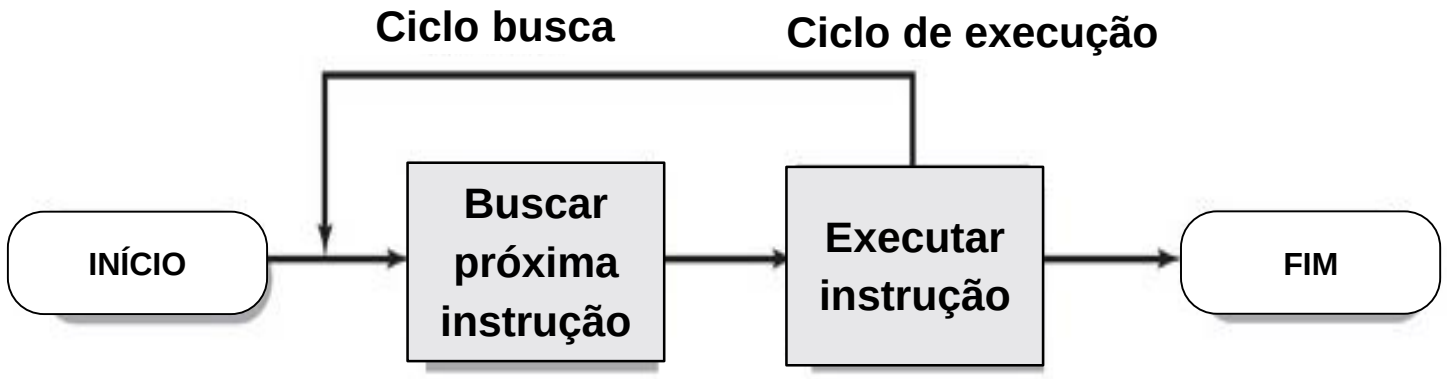
e) A arquitetura Harvard utiliza BIOS diferentes para *input* e *output*.

Ciclo de instrução

- A execução de um programa é baseada em repetir o processo de busca e execução de instruções a serem executadas em um mesmo ciclo.

- A execução de instruções em um ciclo é dependente do tipo de instrução que será executada e, intrinsicamente, de qual operação será realizada.

- Após o término da execução das instruções, o processo é finalizado.



Fonte: STALLINGS (2010, p. 56).

Interrupções

- Todos os computadores modernos oferecem algum mecanismo para que as operações que envolvam alguma comunicação com a memória ou entre a CPU e outros dispositivos de E/S possam ser interrompidos em algum momento de sua operação.

- Embora pareça ao usuário que uma máquina trabalhe melhor sem ser interrompida, as interrupções na verdade garantem uma operação otimizada dos processos.

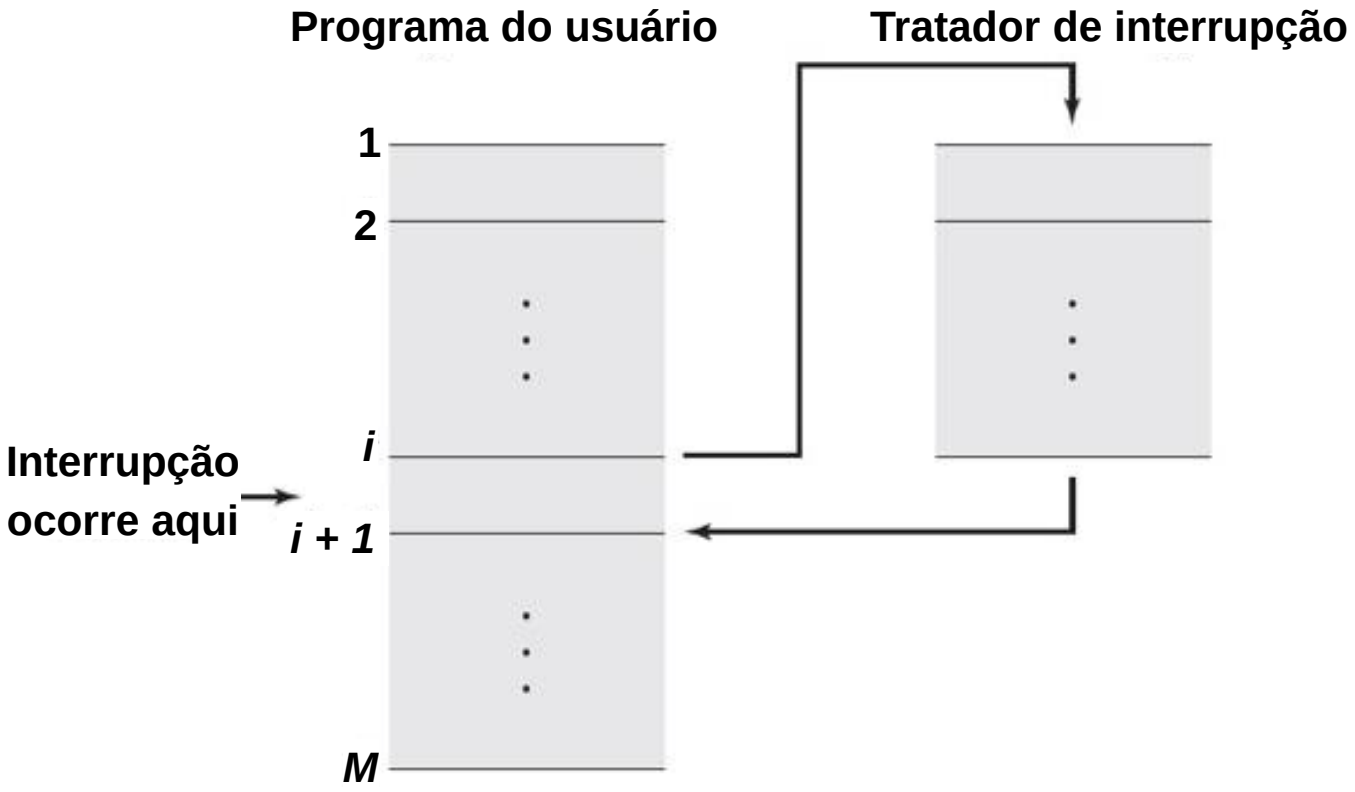
- Uma interrupção nada mais é do que uma quebra na sequência de execução natural de algum *software* ou *hardware*.

    - Quando o tratamento da interrupção tiver terminado, a execução retornará do ponto onde foi interrompido.

Interrupções

- A CPU e o sistema operacional são os responsáveis pela suspensão do programa de usuário para depois retorná-lo ao mesmo ponto.

- Assim, o programa de usuário não precisará conter qualquer outro código especial para acomodar as interrupções.



Fonte: STALLINGS (2010, p. 62).

Interrupções

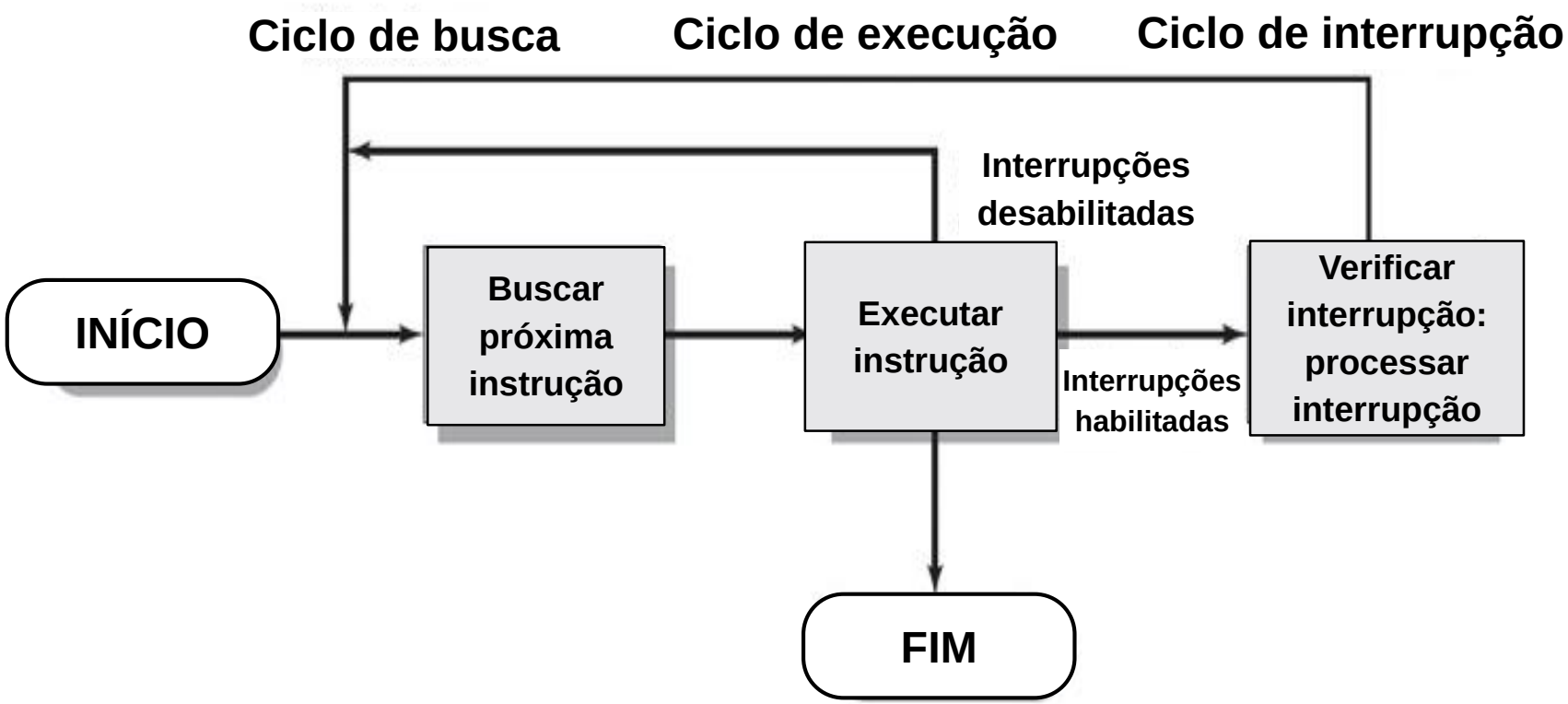
- Para que as interrupções ocorram, um ciclo de interrupção será acrescentado ao ciclo de instrução direto.

- No ciclo de instrução com interrupções, o processador verificará se houve algum sinal de interrupção.

- Se nenhuma interrupção ocorrer ou estiver pendente, o processador pode seguir para a próxima etapa do ciclo de busca, que é ler a próxima instrução do programa em execução.

Interrupções

- Ciclo de instrução com interrupção.



Fonte: STALLINGS (2010, p. 62).

Interrupções múltiplas

Duas técnicas podem ser utilizadas para tratar múltiplas interrupções:

- Na primeira, desativam-se as interrupções enquanto uma interrupção já estiver sendo tratada, o que significa que o processador poderá ignorar qualquer novo sinal de requisição de interrupção.

- Se alguma interrupção ocorrer durante esse período, ela permanecerá pendente e será averiguada pelo processador após ele haver habilitado novamente as interrupções.

Interrupções múltiplas

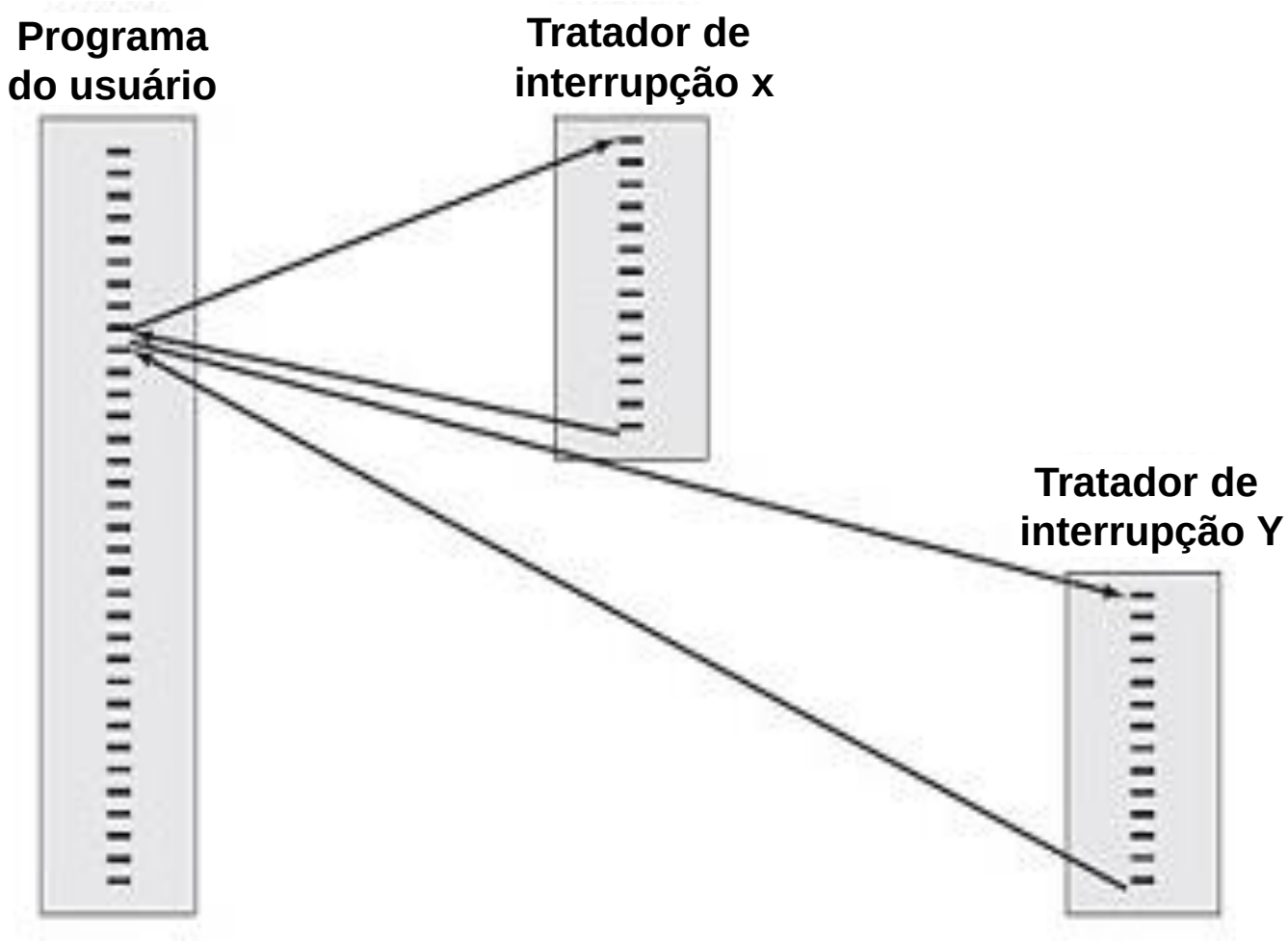
- Depois que a rotina de tratamento de interrupção tiver terminado, as interrupções serão novamente habilitadas antes que o programa de usuário dê continuidade do ponto de parada.

- Dessa forma, o processador verificará se houve interrupções adicionais durante o período em que as interrupções foram desabilitadas.

- Apesar de simples, essa técnica é muito boa, pois as interrupções são tratadas de forma sequencial.

Interrupções múltiplas

- Interrupções múltiplas sequenciais.

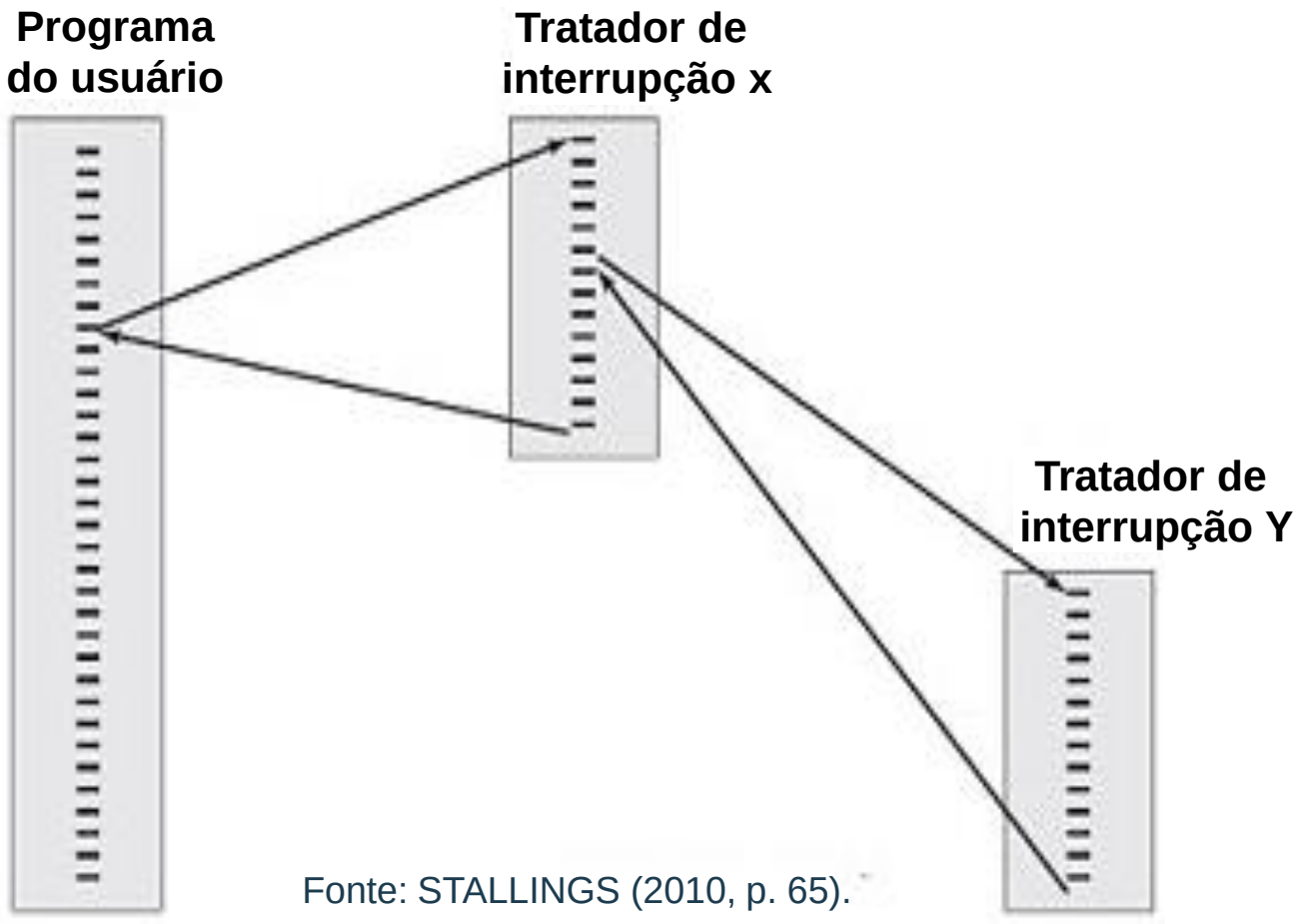


Fonte: STALLINGS (2010, p. 65).

Interrupções múltiplas

- Em outra técnica, definem-se as prioridades de interrupção a fim de permitir que uma interrupção de maior prioridade possa ter um tratamento de interrupção distinto ao de menor prioridade, inclusive ocasionando a pausa desse com menor prioridade.

- Interrupções múltiplas aninhadas.



Fonte: STALLINGS (2010, p. 65).

Interatividade

Uma interrupção pode ser definida, de forma simplificada, como uma quebra na sequência de execução normal de algum *software* ou *hardware*, que deverá transferir o controle das tarefas através de interrupções para realizar qual tarefa?

a) Armazenar dados.

b) Tratar a interrupção.

c) Buscar instrução anterior.

d) Finalizar a tarefa posterior.

e) Armazenar instruções.

Resposta

Uma interrupção pode ser definida, de forma simplificada, como uma quebra na sequência de execução normal de algum *software* ou *hardware*, que deverá transferir o controle das tarefas através de interrupções para realizar qual tarefa?

a) Armazenar dados.

b) Tratar a interrupção.

c) Buscar instrução anterior.

d) Finalizar a tarefa posterior.

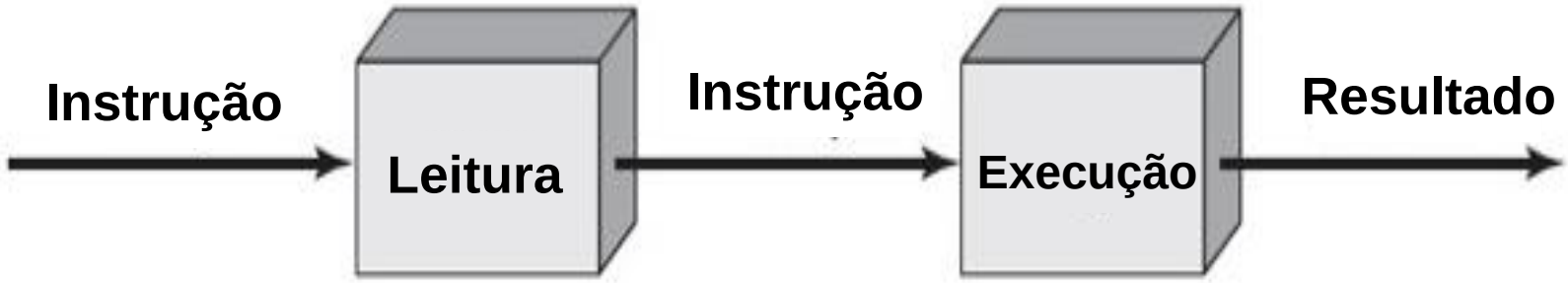
e) Armazenar instruções.

Pipeline de instruções

- O paralelismo em nível de instrução ou *pipeline* busca essencialmente sobrepor a execução de instruções a fim de melhorar o desempenho de processamento.

- Uma instrução em um computador possui vários estágios operacionais, que vão desde o início da tarefa, até sua conclusão.

- Essas tarefas ocorrem, geralmente, de forma sequencial, ou seja, uma após a outra.



Fonte: STALLINGS (2010, p. 365).

Pipeline de instruções

- O processo de *pipelining* pode acelerar a execução das instruções se os estágios de leitura e execução tiverem o mesmo tempo de duração.

- Assim, o ciclo de instrução poderá ser reduzido pela metade do tempo total de execução.

- Entretanto alguns fatores impossibilitam essa melhora no desempenho, como:

    - O tempo de execução da instrução obviamente é maior do que o tempo de leitura.

    - Uma instrução de desvio pode ocorrer durante a execução da instrução, o que faz com que o endereço da próxima instrução não seja conhecido.

Pipeline de instruções

- Mesmo com esses problemas, algum ganho na aceleração do processamento ainda ocorrerá de forma que, para obter mais velocidade, o *pipeline* deverá conter mais estágios a fim de tratar possíveis atrasos em estágios de forma individual.

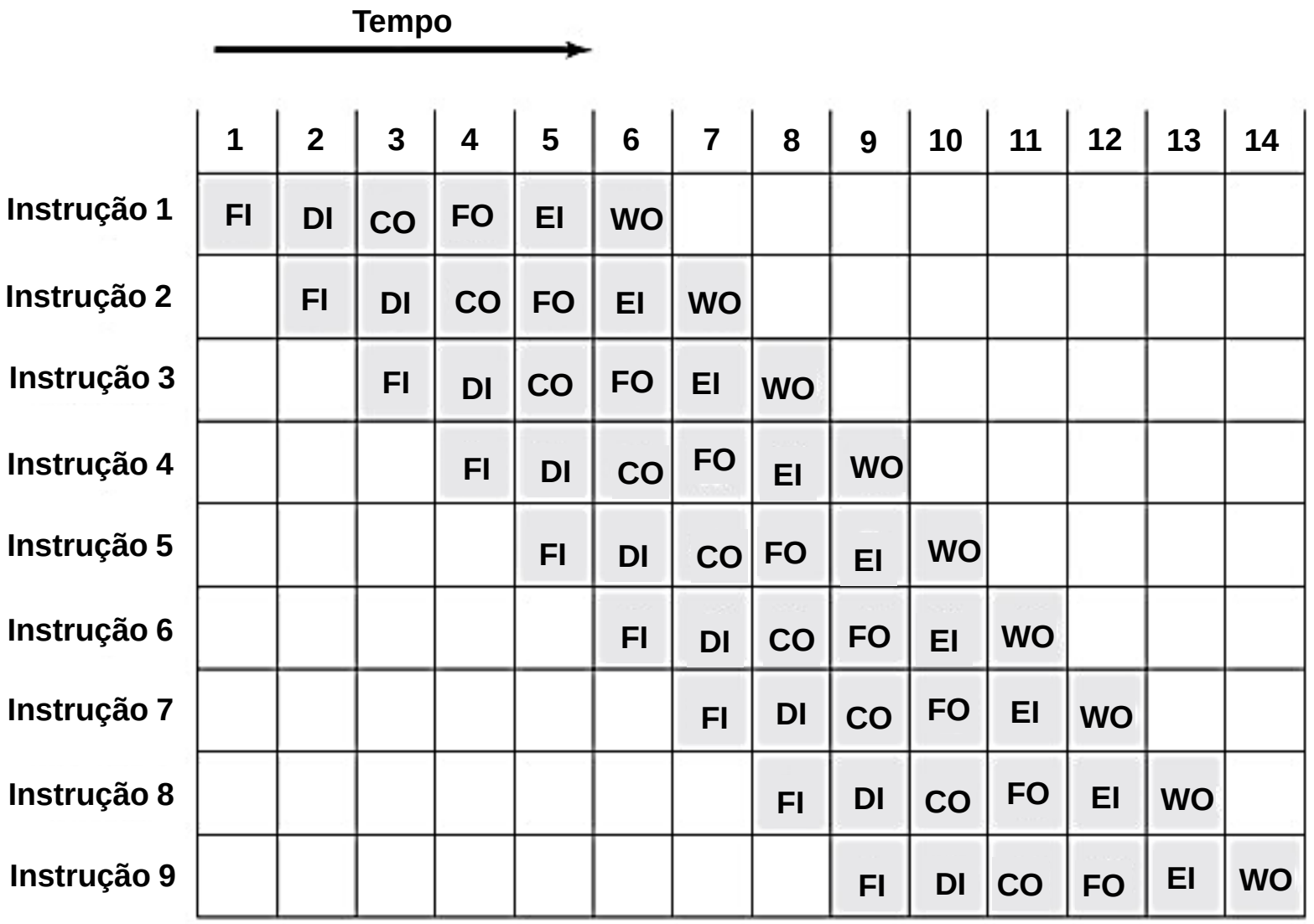
- Para que o *pipeline* obtenha maior desempenho, mais estágios serão necessários, de forma que os estágios possam ser decompostos e com menos operações em cada ciclo.

- Um sistema de *pipeline* pode conter 6 estágios, divididos em:

    - Buscar instrução (FI – *fetch instruction*), decodificar instrução (DI – *decode instruction*), calcular operandos (CO – *calculate operand*), obter operandos (FO – *fetch operands*), executar a instrução (EI – *instruction execution*), escrever operando (WO – *write operands*).

Pipeline de instruções

- *Pipeline* sem interrupção.



Fonte: STALLINGS (2010, p. 366).

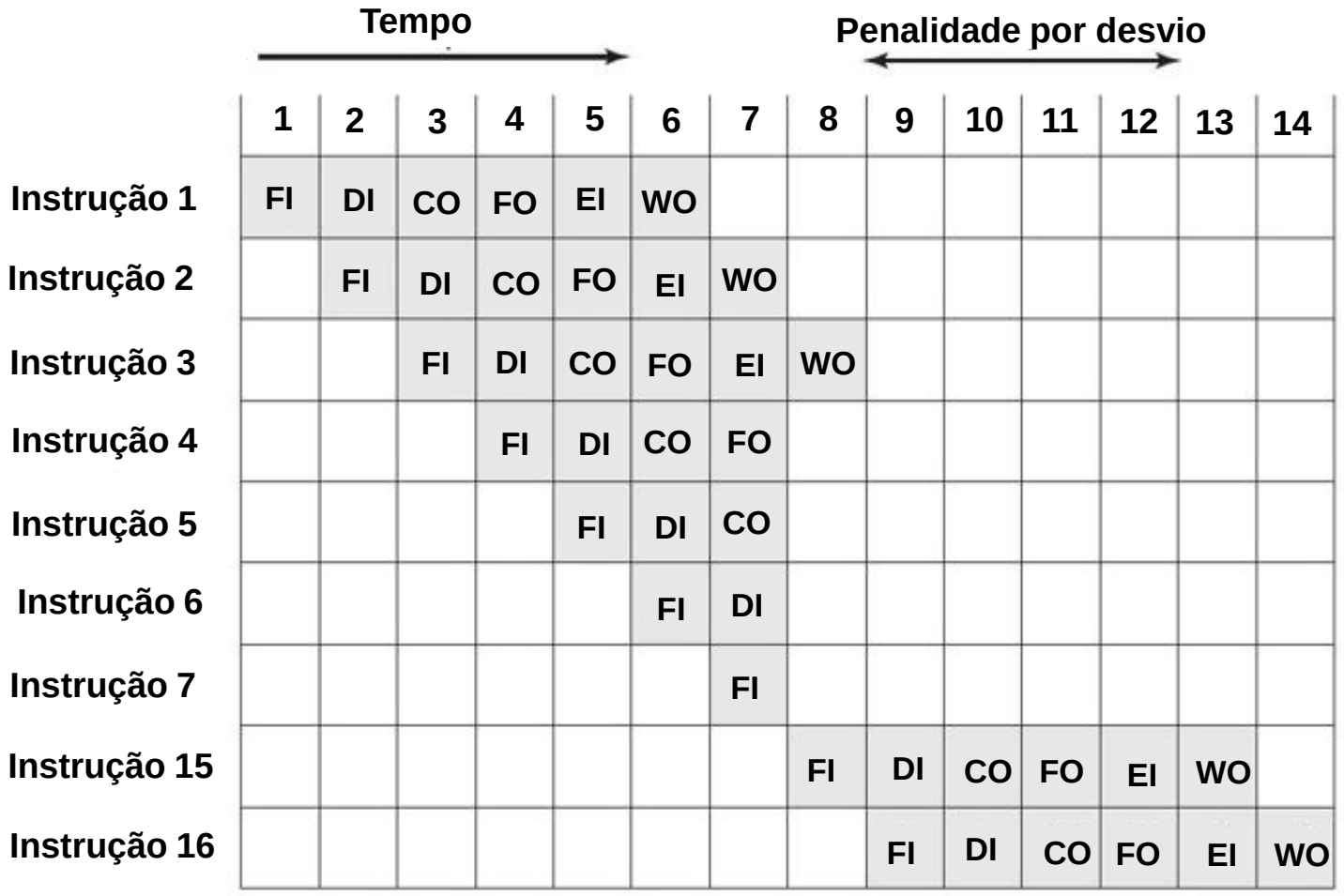
Pipeline de instruções

- Os desvios caracterizam uma penalidade no desempenho no *pipeline*.
- Pois é de interesse que, durante todo o processo, seja possível antecipar o desvio, tratando-o como uma interrupção e depois retornar ao ponto da parada do processo.
- O simples fato de limpar as instruções do *pipeline* faz com que os dados e as instruções já carregados no processo sejam perdidos.

Pipeline de instruções

- Em uma situação real, os 6 estágios não terão a mesma duração, ocasionando espera entre os estágios. Uma instrução de desvio condicional, por exemplo, pode invalidar várias leituras de instruções.

- *Pipeline* com interrupção.



Fonte: STALLINGS (2010, p. 367).

Hazard de *Pipeline*

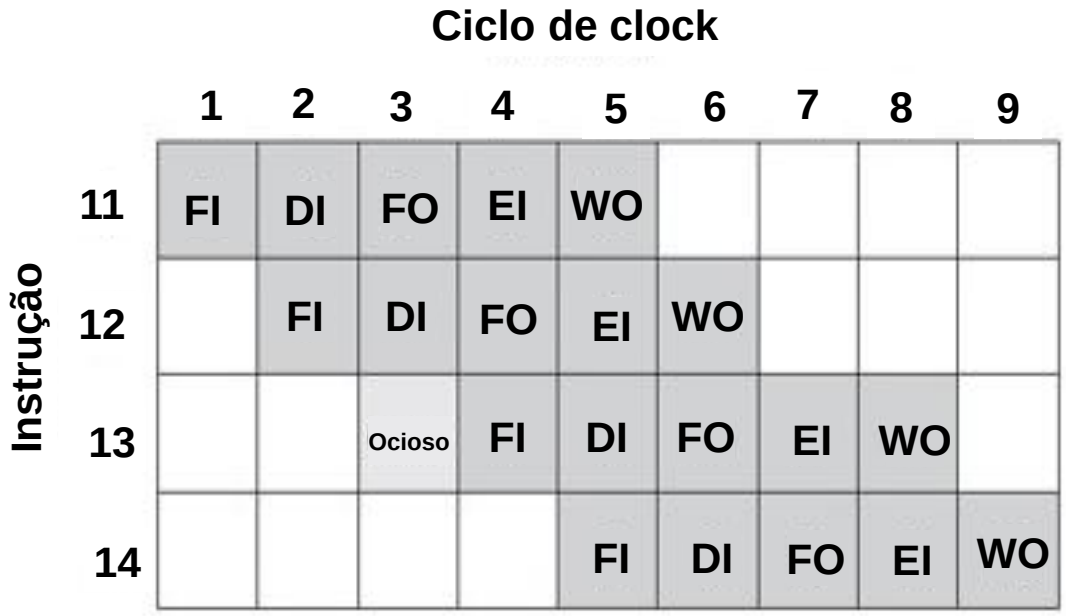
- Um *hazard* de *pipeline* ou bolha de *pipeline* é definido como um evento em que a próxima instrução de *pipeline* não poderá ser executada no ciclo de *clock* seguinte, ocasionando um efeito bolha no *pipeline*.

- Isso ocorre no *pipeline* devido a alguma parte dele precisar parar de executar por causas que não permitem sua execução contínua, como um desvio, por exemplo.

- Os *hazards* de *pipeline* são subdivididos em três categorias: *hazards* de recursos, *hazards* de dados e *hazards* de controle.

Hazard de *Pipeline*

- *Hazards* de recursos: também conhecido como *hazard* estrutural, ocorre quando duas ou mais instruções, que já estão no *pipeline*, necessitam do mesmo recurso, resultando que as instruções precisarão ser executadas em série em vez de paralelo.



Fonte: STALLINGS (2010, p. 371).

Hazard de *Pipeline*

- *Hazards* de dados: uma bolha de dados irá ocorrer quando há um conflito no acesso de uma posição na memória de algum dado ou operando.
- Duas instruções de um programa estão a ponto de serem executadas na sequência, de modo que ambas as instruções acessam um determinado operando, localizado ou na memória ou em algum registrador, ao mesmo tempo.

Ciclo de *clock*

1	2	3	4	5	6	7	8	9	10
FI	DI	FO	EI	EO					
	FI	DI	Ocioso		FO	EI	WO		
		FI			DI	FO	EI	WO	
					FI	DI	FO	EI	WO

Fonte: STALLINGS (2010, p. 372).

Hazard de *Pipeline*

- *Hazards* de controle: uma bolha de controle, conhecido também como *hazard* de desvio, ocorre quando o *pipeline* toma uma decisão errada ao prever algum tipo de desvio.

- Dessa forma, traz instruções para dentro do *pipeline,* que precisarão ser descartadas logo após terem sido adicionadas ao *pipeline*.

Interatividade

Um *hazard,* ou bolha de *pipeline,* é definido como um evento em que a próxima instrução de *pipeline* não poderá ser executada no ciclo de *clock* seguinte, ocasionando um atraso na execução dos processos. Isso ocorre devido a alguma parte dele precisar parar de executar sua tarefa original e ter que atender a algum desvio. Os *hazards* de *pipeline* são subdivididos geralmente em quais categorias?

a) *Hazard* de instrução, *hazard* de tarefas e *hazard* de memória.
b) *Hazard* de dados, *hazard* de instruções e *hazard* de controle.
c) *Hazard* de análise, *hazard* de memória e *hazard* de dados.
d) *Hazard* de recursos, *hazard* de dados e *hazard* de controle.
e) *Hazard* de algoritmo, *hazard* de tarefas e *hazard* de dados.

Resposta

Um *hazard,* ou bolha de *pipeline,* é definido como um evento em que a próxima instrução de *pipeline* não poderá ser executada no ciclo de *clock* seguinte, ocasionando um atraso na execução dos processos. Isso ocorre devido a alguma parte dele precisar parar de executar sua tarefa original e ter que atender a algum desvio. Os *hazards* de *pipeline* são subdivididos geralmente em quais categorias?

a) *Hazard* de instrução, *hazard* de tarefas e *hazard* de memória.

b) *Hazard* de dados, *hazard* de instruções e *hazard* de controle.

c) *Hazard* de análise, *hazard* de memória e *hazard* de dados.

d) *Hazard* de recursos, *hazard* de dados e *hazard* de controle.

e) *Hazard* de algoritmo, *hazard* de tarefas e *hazard* de dados.

ATÉ A PRÓXIMA!