Organização do Processador: Pipeline

Arquitetura e Organização de Computadores

Prof. Lucas de Oliveira Teixeira

UEM

Pipeline

Pipeline

- É uma técnica de implementação de processadores que permite a sobreposição temporal das diversas fases de execução das instruções.
- É semelhante a uma linha de montagem industrial, onde produtos em vários estágios podem ser trabalhados simultaneamente.
- Aumenta o número de instruções executadas simultaneamente. Aumenta o chamado de throughput.
- Importante: O pipeline não reduz o tempo gasto para completar cada instrução individualmente.

Pipeline

Passos para execução de uma Instrução:

- Buscar instrução (FI – Fetch Instruction).
- Decodificar instrução (DI – Decode Instruction).
- Calcular operandos (CO – Calculate Operands).
- Buscar operandos (FO – Fetch Operands).
- Executar instrução (EI – Execute Instruction).
- Escrever resultado (WO – Write Operand).

A ideia é sobrepor estas operações!

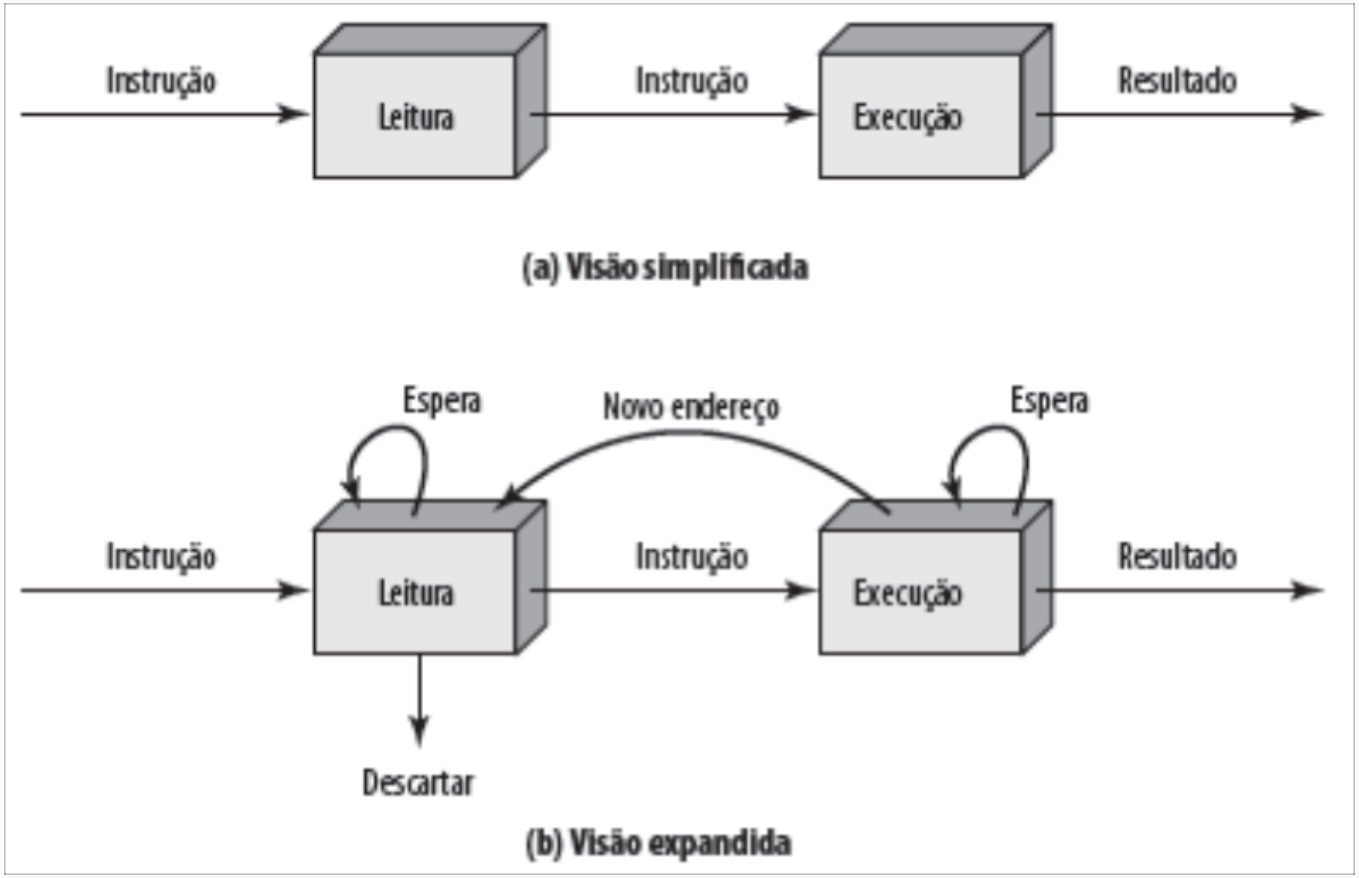
Pipeline

Analogia:

- Linha de produção de uma montadora de carros.
- Processo de lavagem de roupas (lavar, secar, passar, arrumar).

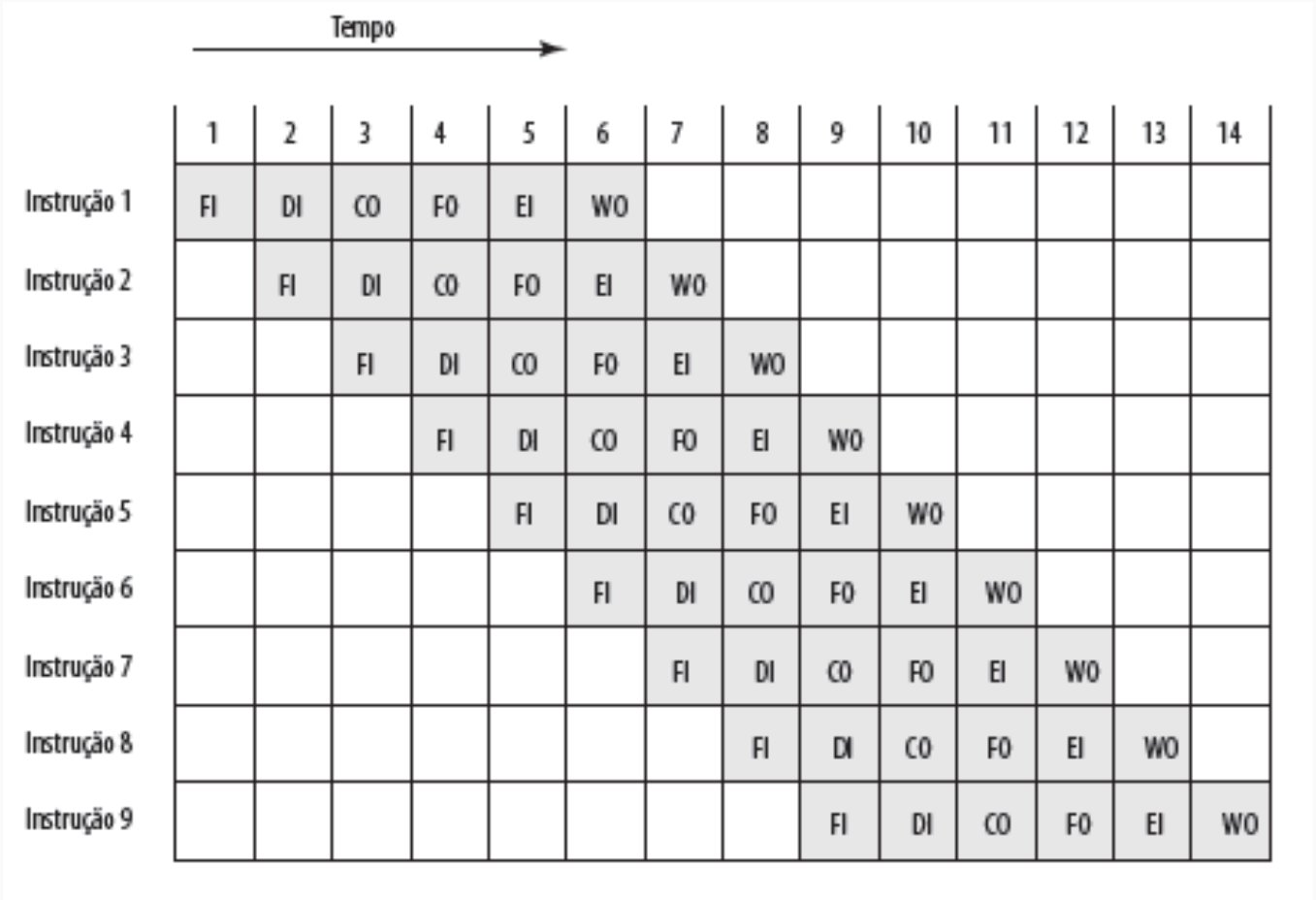
Pipeline

Pipeline de dois estágios:



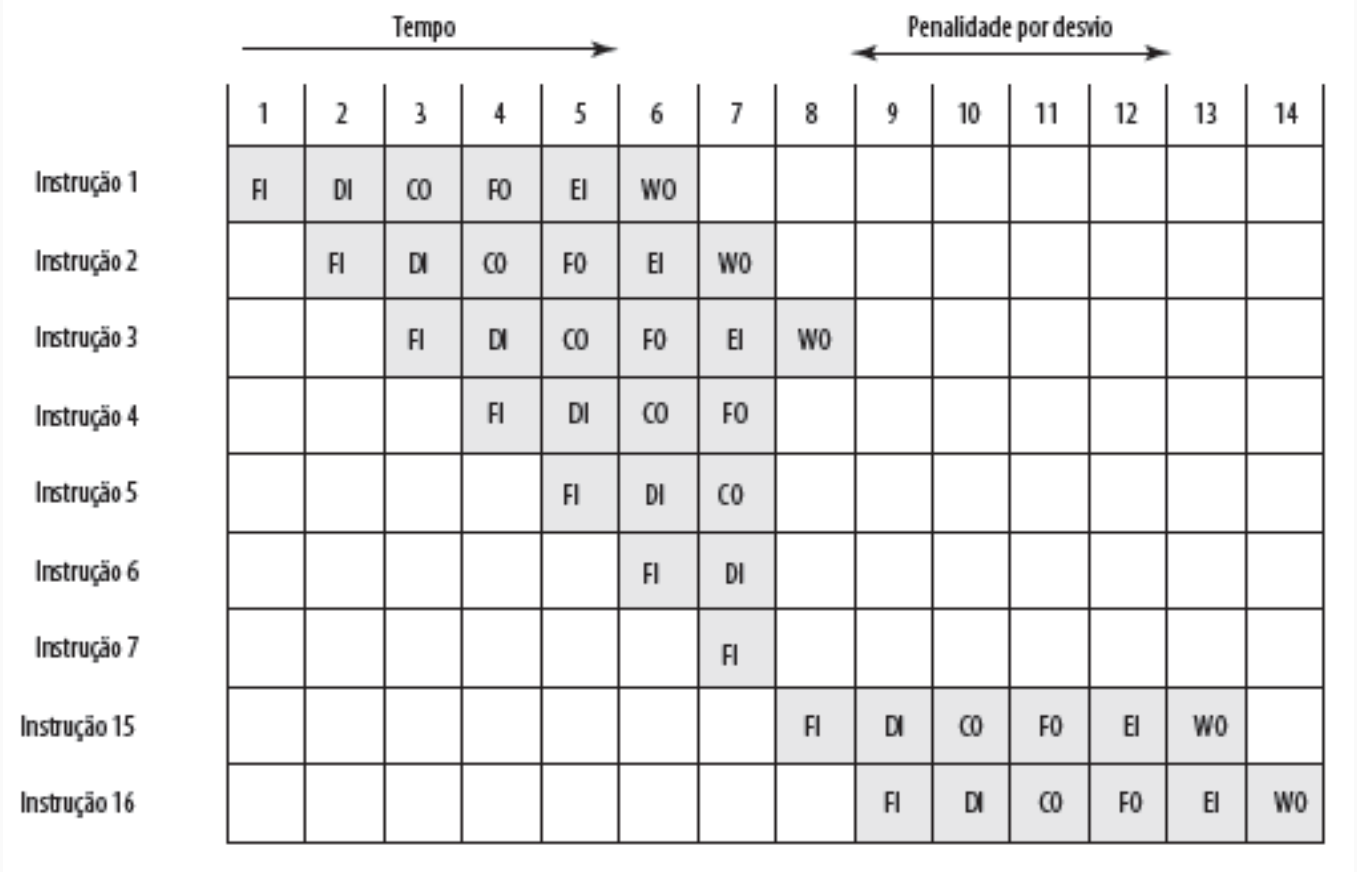
Pipeline

Pipeline de seis estágios ao longo do tempo:



Pipeline

Desvio condicional:



Hazard

Hazard

- O hazard é algum problema com o fluxo de instruções no pipeline, assim ele precisa parar totalmente ou em partes.
- Também conhecido como bolha no pipeline.
- Tipos de hazards: recursos, dados e controle.

Hazard

Hazard de recursos:

- Duas (ou mais) instruções no pipeline precisam do mesmo recurso.
- Também chamado de hazard estrutural.

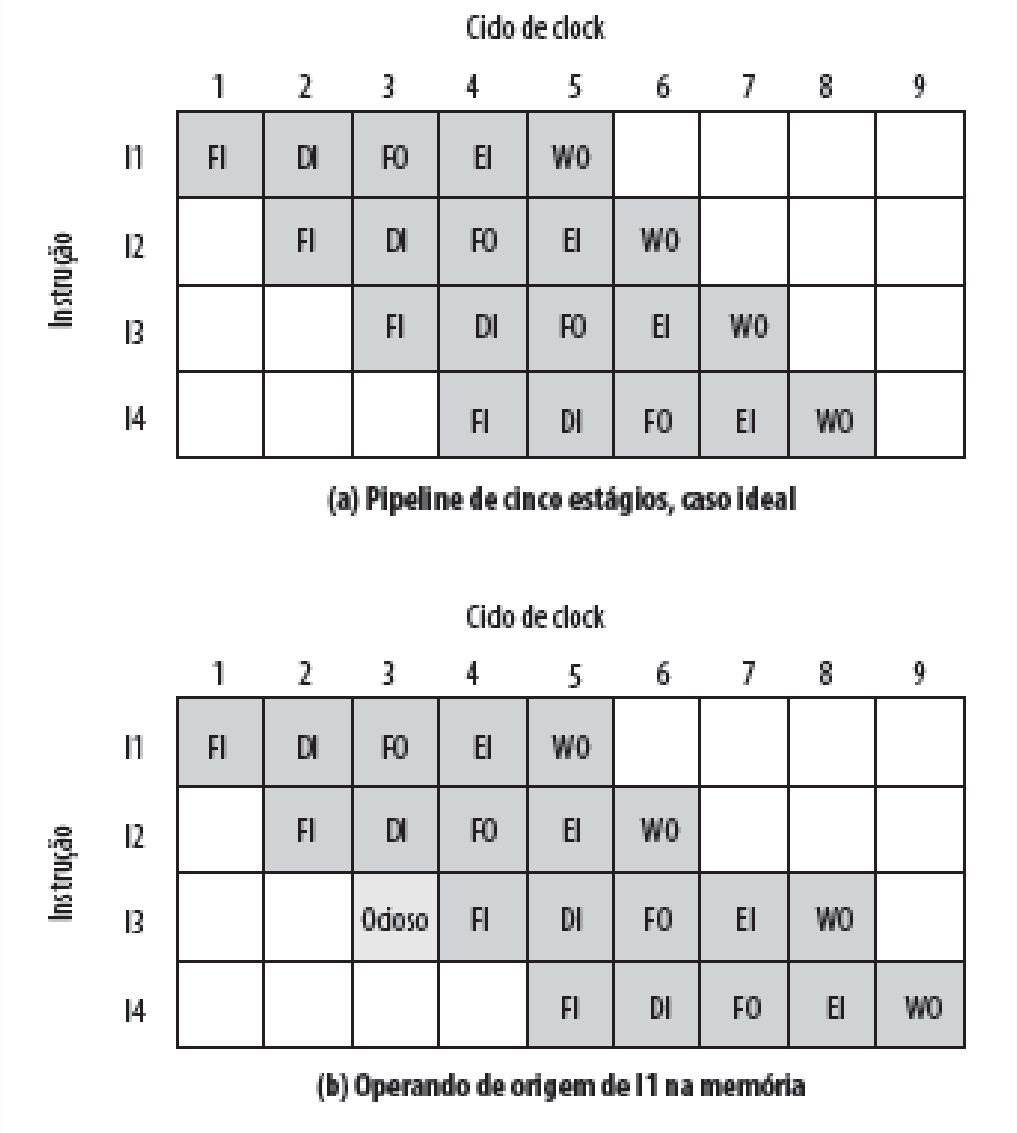
Hazard

Hazard de recursos:

- Considere um pipeline de 5 estágios com cada estágio usa um ciclo de clock.
- No caso ideal, a cada ciclo de clock uma nova instrução entra no pipeline.
- Suponha que a memória principal tenha uma única porta para comunicação.
- A busca de instrução, e leitura/escrita de dados devem ser feitas uma por vez.
- O problema é que a leitura/escrita de operando não podem ser realizadas em paralelo com busca de instrução.

Hazard

Hazard de recursos:



Hazard

Hazard de recursos:

- Estágio de busca de instrução fica ocioso por um ciclo buscando I3.
- Uma solução é aumentar recursos disponíveis. Neste caso, aumentar as portas de acesso a memória principal.

Hazard

Hazard de dados:

- Conflito no acesso de um local de operando.
- Duas ou mais instruções executadas em sequência acessando uma região de memória em particular ou registrador.
- Como o resultado da primeira instrução só é salvo no final do pipeline, as instruções subsequentes acabam acessando valores antigos.

Hazard

Hazard de dados:

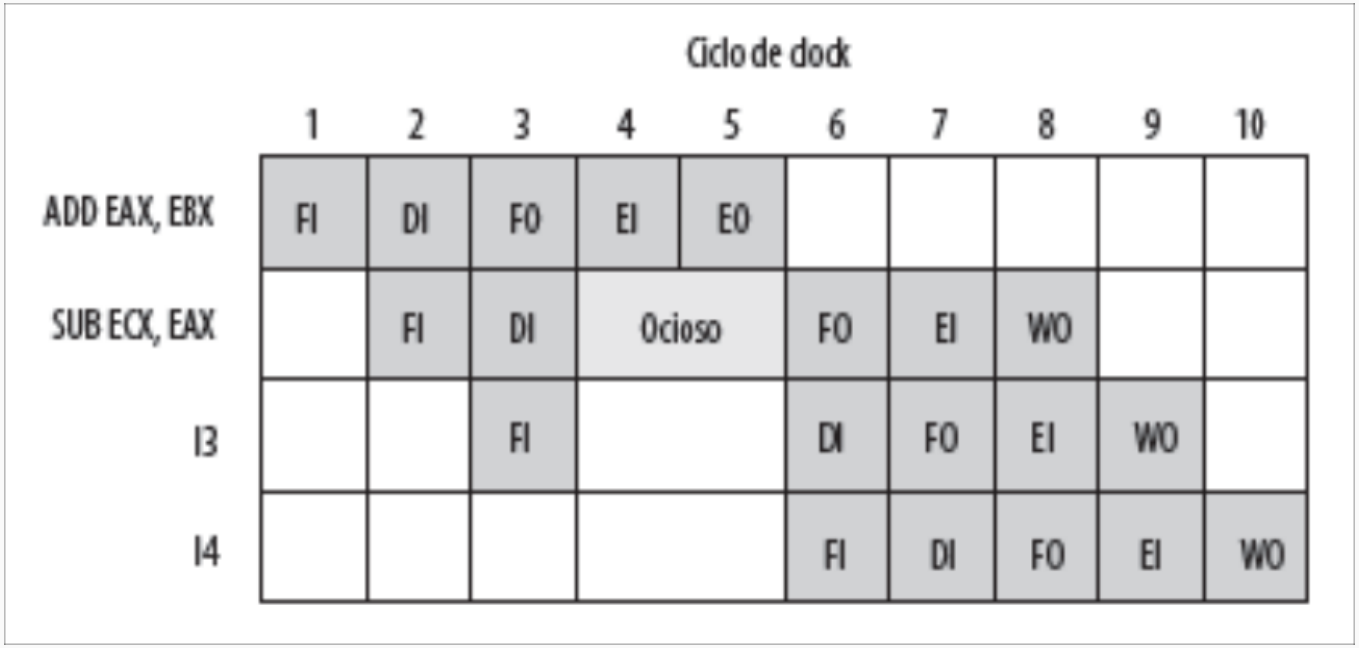
- Considerando a sequência de instruções:

```
1 add rax, rbx
2 sub rcx, rax
```

- A instrução ADD não atualiza RAX até o final da escrita dos resultados.
- A instrução SUB precisa esperar dois ciclos de clock até que o valor esteja atualizado.

Hazard

Hazard de dados:



Hazard

Leitura após escrita (RAW):

- Chamado de dependência verdadeira.
- Uma instrução modifica um registrador ou local de memória.
- Instrução seguinte lê dados nesse local.
- O hazard ocorre se leitura for antes do término da escrita.
- Por exemplo:

```
1 add rax, rbx
2 sub rcx, rax
```

Hazard

Escrita após leitura (WAR):

- Chamado de antidependência.
- Uma instrução lê um registrador ou local da memória.
- Instrução seguinte escreve no local.
- O hazard ocorre se escrita termina antes que ocorra a leitura (processadores superescalares, execução de mais de uma instrução por ciclo de clock).
- Por exemplo:

```
1 add rax, rbx
2 sub rbx, rcx
```

Hazard

Escrita após escrita (WAW):

- Chamado de dependência de saída.
- Duas instruções escrevem no mesmo local.
- O hazard se a escrita ocorre na ordem contrária à sequência intencionada.
- Por exemplo:

```
1 add r3, r2, r1
2 sub r3, r4, r5
```

Hazard

Técnicas de solução de hazard de dados:

- Forwarding: Os resultados da execução são enviados para a busca de operandos para evitar a dependência de dados
- Reordenação de código: Um compilador inteligente reordena o código em Assembly para evitar esses tipos de dependência.
- Inserção de bolhas: Inserção proposital de uma bolha no pipeline para atrasar a execução da instrução, ganhando tempo para a instrução posterior. Uso de instruções NOOP (No Operation).

Hazard

Hazard de controle:

- Conhecido como hazard de desvio.
- Neste caso, o pipeline toma uma decisão errada sobre a previsão de desvio.
- Desvio é um jump condicional (JZ, JN, etc...).
- O pipeline fica com instruções que precisam ser descartadas subsequentemente.
- Existem diversas técnicas para resolver tal problema.

Hazard

Múltiplos fluxos de execução:

- Possuir dois pipelines.
- Buscar antecipadamente cada instrução em (com desvio sendo tomado ou não) um pipeline separado.
- Após desvio ser descoberto, o pipeline apropriado é usado.
- O problema é que ocasiona disputa por barramento e registradores. Ex.: Os dois lados querem modificar o mesmo registrador.

Hazard

Busca antecipada do alvo:

- Instrução alvo do desvio é buscada antecipadamente além das instruções após o desvio.
- Mantém instrução alvo até que o desvio seja executado.
- Caso o desvio seja tomado, a instrução alvo do desvio é inserida no pipeline.
- Caso contrário, segue o jogo.

Hazard

Buffer de laço de repetição:

- Um buffer é uma memória muito rápida.
- Mantido pelo estágio de busca do pipeline.
- Guarda n instruções buscadas mais recentemente, em sequência.
- Verificar buffer antes de buscar próxima instrução na memória.
- Geralmente é usado em conjunto com a busca antecipada.
- Muito bom para laços pequenos.

Hazard

Previsão de desvios:

- O pipeline tenta prever se um desvio condicional será tomado ou não.
- Existem várias técnicas de previsão de desvio.

Hazard

Previsão de desvios estática:

- Previsão nunca tomada: Assume que salto nunca acontecerá e sempre busca próxima instrução.
- Previsão sempre tomada: Assume que salto sempre acontecerá e sempre busca instrução alvo.

Hazard

Previsão de desvios por opcode:

- Algumas instruções (JZ, JE, JN, etc...) são mais prováveis de resultar em um salto do que outras.
- Probabilidade calculada estatisticamente.
- Pode chegar em até 75% de sucesso.

Hazard

Previsão de desvios por histórico:

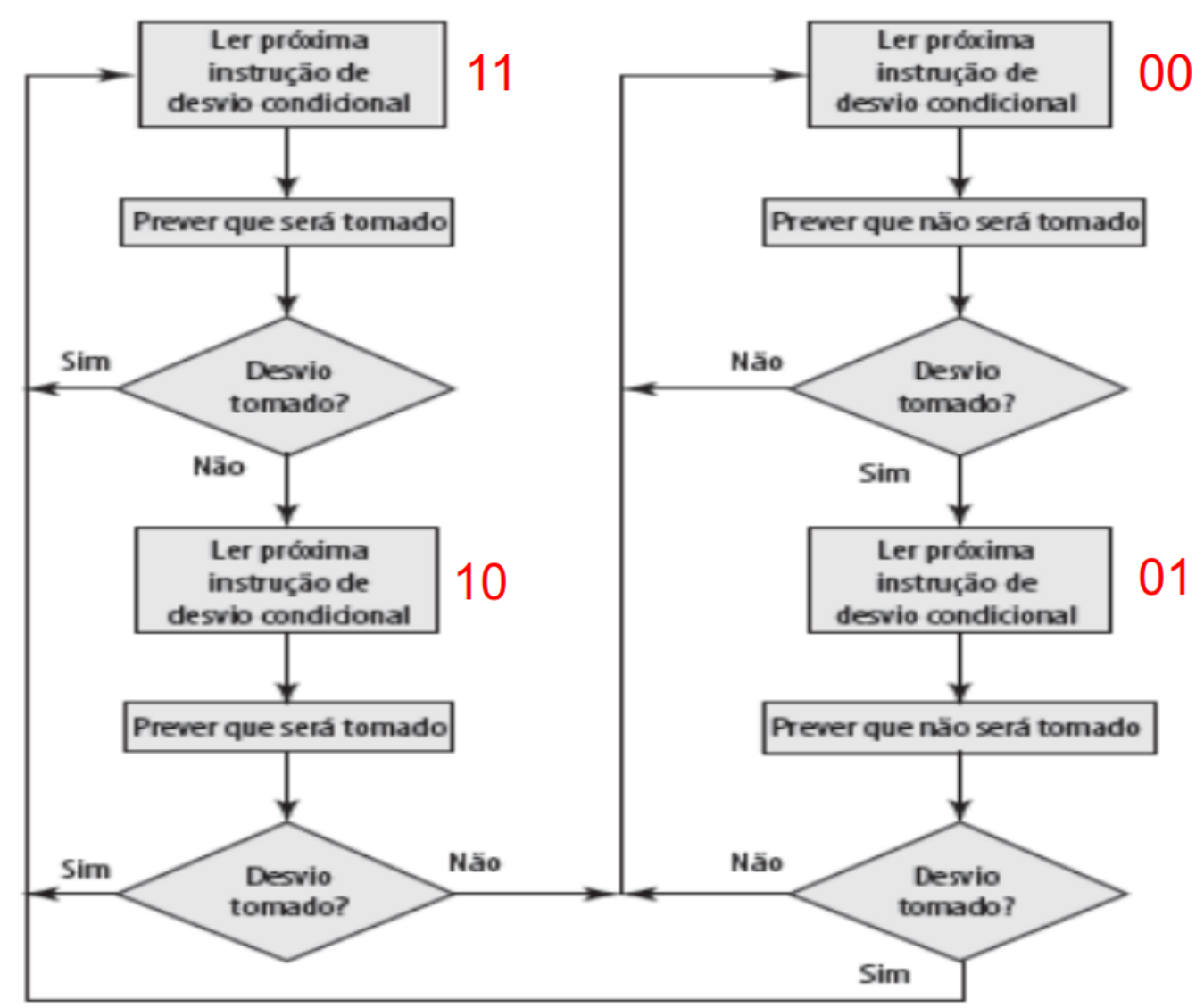
- Baseada no histórico do desvio.
- Um ou mais bits podem ser associados com cada instrução de desvio condicional que reflete o seu histórico recente.
- Geralmente 1 ou 2 bits são utilizados:
  - 1 Bit: Sabe-se apenas sobre a última execução do desvio.
  - 2 Bits: Sabe-se sobre as duas últimas execuções do desvio.
- Boa para laços (Executa várias vezes o mesmo desvio).
- Diferente das técnicas anteriores, é uma abordagem dinâmica.

Hazard

Previsão de desvios por histórico com 2 bits:

- 00 - Nas duas últimas execuções não foi tomado.
- 01 - Na última foi tomado e na penúltima não.
- 10 - Na última não foi tomado e na penúltima foi.
- 11 - Nas duas últimas execuções foram tomados.

Previsão de desvios por histórico com 2 bits:



Hazard

Tabela de histórico de desvio:

- Os últimos resultados de desvios individuais são armazenados em uma tabela associativa.
- Assim, o histórico é individual para cada instrução de desvio.
- As vezes, pode considerar também o comportamento das instruções de desvios próximas à instrução alvo.

Hazard

Desvio atrasado:

- Não salta até que você realmente precise.
- Reorganiza as instruções.
- Veremos quando falarmos mais sobre arquiteturas RISC.