INE5607 – Organização e Arquitetura de Computadores

Unidade Central de Processamento

Aula 16: Processadores pipeline

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br

Sumário

- Paralelismo em nível de instrução
- Pipelining
- Hazards
- Data hazards e forwarding
- Control hazards e predição de desvios
- Considerações finais

PARALELISMO EM NÍVEL DE INSTRUÇÃO

Paralelismo em nível de instrução

- **Problemas de processadores monociclo**
  - Uma instrução por ciclo
  - Período do ciclo de relógio depende da instrução mais lenta
    - Instruções muito lentas -> processadores muito lentos
    - Sofre com o tempo de acesso à memória (duas vezes!)

Paralelismo em nível de instrução

- **Solução prática**
  - Ninguém* fabrica processadores monociclo
  - Ninguém executa apenas uma [parte de] instrução de cada vez
    - A visão de um programa sequencial é preservada
      - Programamos instrução após instrução
      - Execução real pode ser um pouco diferente

*depende de verificação

Paralelismo em nível de instrução

- **Paralelismo em nível de instrução**
  - ILP: *instruction level parallelism*
  - Execução paralela de instruções ou partes de instruções de um mesmo fluxo
    - Diferente de executar programas diferentes em máquinas diferentes, por exemplo

PIPELINING

Pipelining

- Analogia com lavagem de roupas

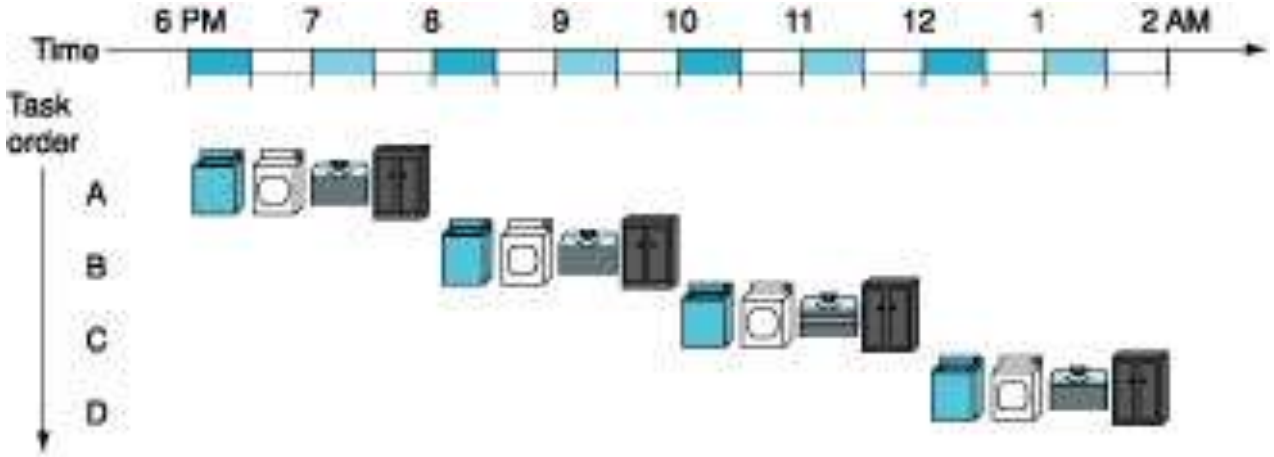


Figura 4.25 do livro *Computer Organization and Design 4th ed.*

Pipelining

- Analogia com lavagem de roupas

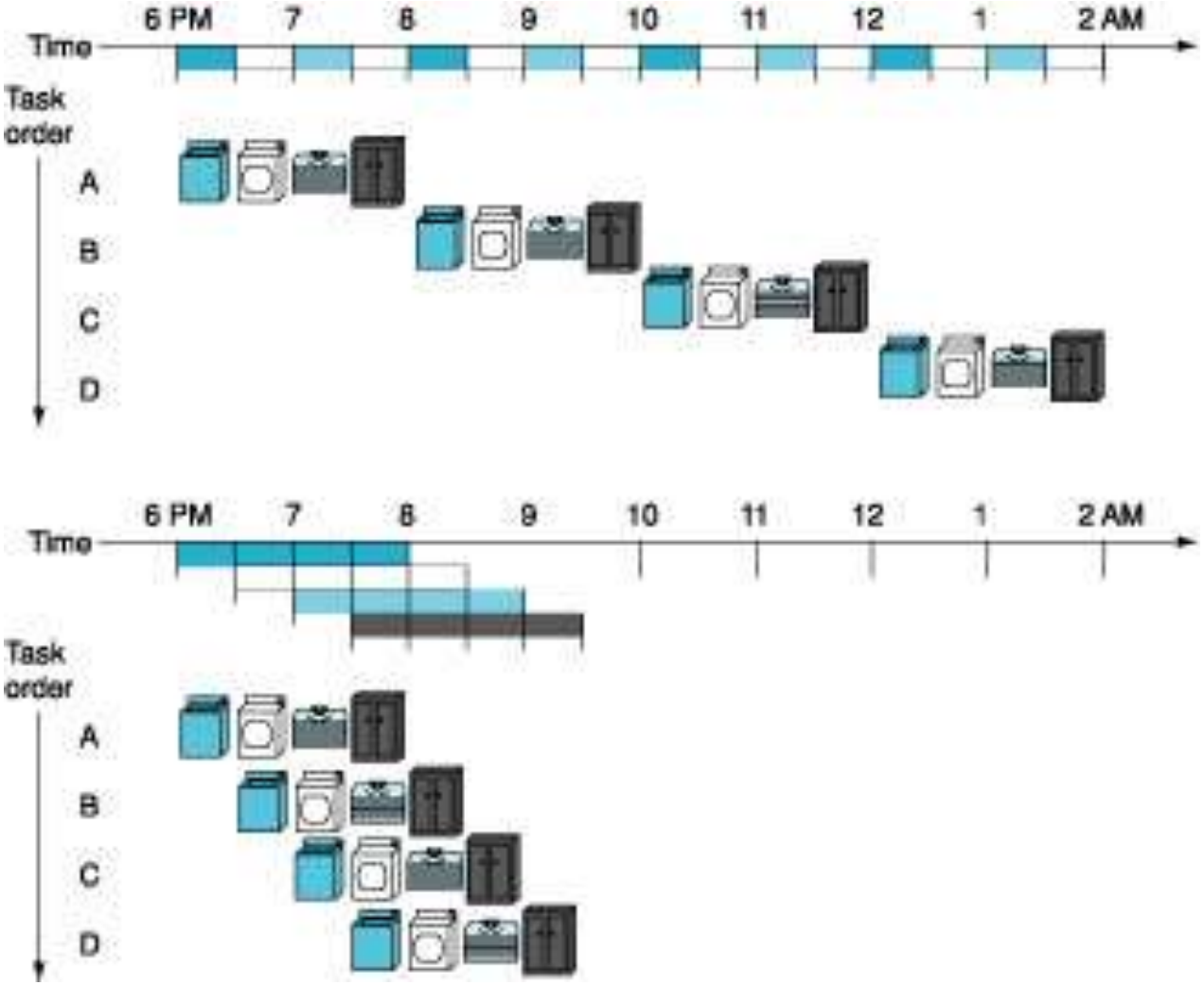


Figura 4.25 do livro *Computer Organization and Design 4th ed.*

Pipelining

- Tempo de lavagem de roupa
  - Em sequência: 8 horas
  - Pipeline: 3,5 horas
    - 2,3 vezes mais rápido
    - Diferença tende a quatro
      - Número de estágios

Pipelining

- Pipelining
  - **Técnica de sobreposição de instruções em execução**
  - Cada instrução utiliza componentes diferentes em um dado ciclo
    - Uma instrução é lida, outra usa a ULA, outra escreve na memória, etc.

Pipelining

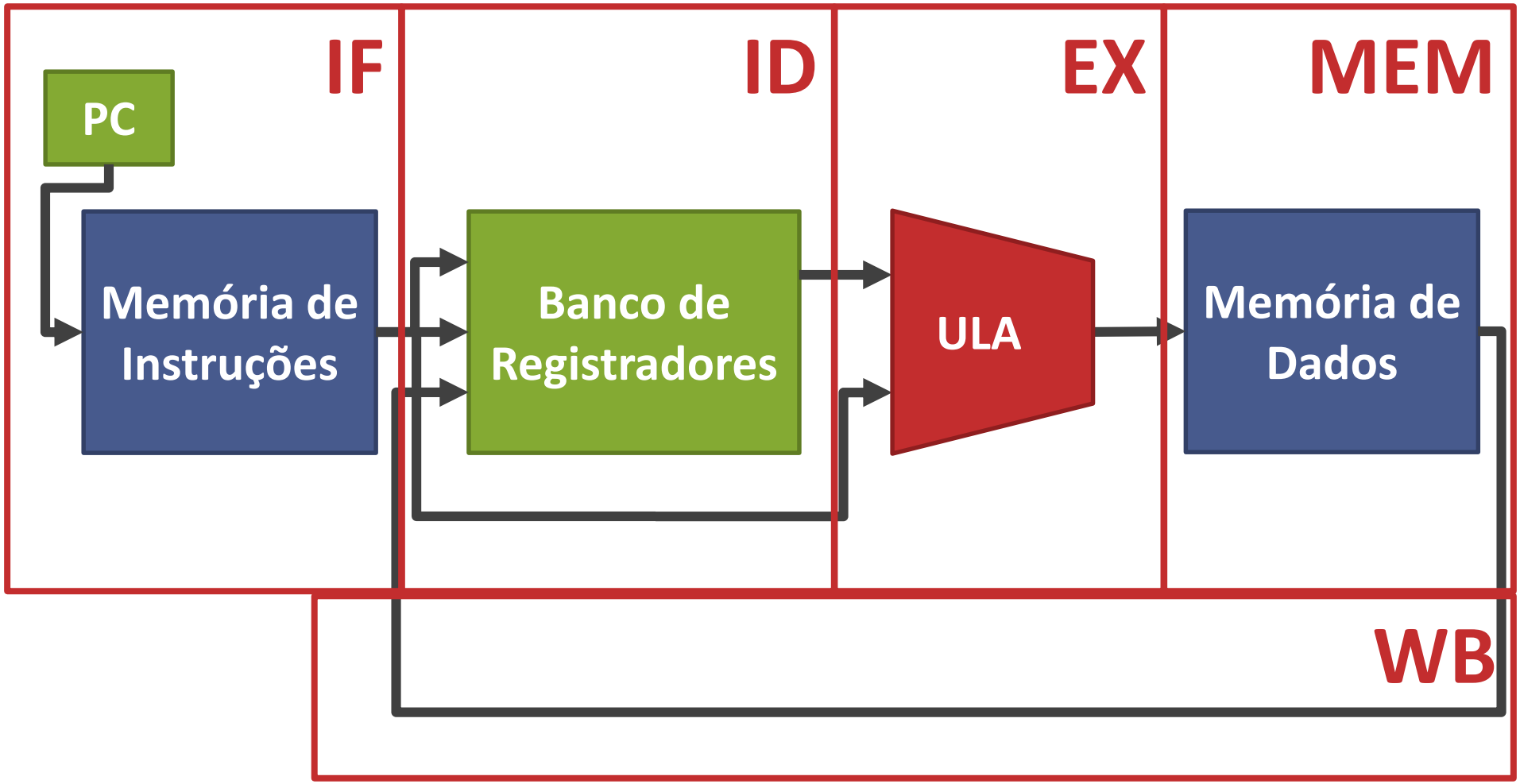
- Pipelining
  - **Não muda o tempo de execução de uma instrução**
    - Não muda a latência de uma instrução
  - **Aumenta a vazão de instruções**
    - Mais instruções em um dado período de tempo
  - **Aceleração potencial igual ao número de estágios do pipeline**

Pipelining

- Etapas de uma instrução MIPS
  1. Buscar a instrução na memória (**IF**)
  2. Decodificação; leitura de registradores (**ID**)
  3. Execução: operação/cálculo de endereço (**EX**)
  4. Acesso a operando em memória (**MEM**)
  5. Escrita do resultado em registrador (**WB**)

Pipelining

- Etapas para uma instrução de *load word*



Pipelining

- Visualização da execução de um pipeline

Inst\Ciclos	1	2	3	4	5	6	7	8	9
I1	IF	ID	EX	MEM	WB				
I2		IF	ID	EX	MEM	WB			
I3			IF	ID	EX	MEM	WB		
I4				IF	ID	EX	MEM	WB	
I5					IF	ID	EX	MEM	WB
I6						IF	ID	EX	MEM
I7							IF	ID	EX

Pipelining

- Visualização da execução de um pipeline

Inst\Ciclos	1	2	3	4	5	6	7	8	9
I1	IF	ID	EX	MEM	WB				
I2		IF	ID	EX	MEM	WB			
I3			IF	ID	EX	MEM	WB		
I4				IF	ID	EX	MEM	WB	
I5					IF	ID	EX	MEM	WB
I6			Pipeline cheio			IF	ID	EX	MEM
I7							IF	ID	EX

Pipelining

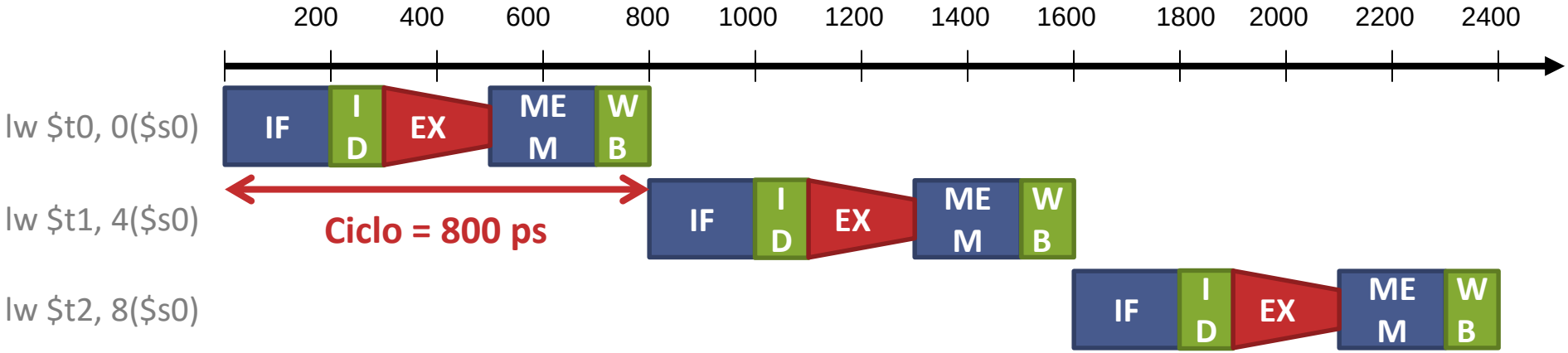
- Comparação monociclo e pipeline

Classe de instrução	Busca da instrução	Leitura do banco de registradores	ULA	Acesso à memória de dados	Escrita no banco de registradores	Total
lw	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
sw	200 ps	100 ps	200 ps	200 ps		700 ps
Tipo R	200 ps	100 ps	200 ps		100 ps	600 ps
beq	200 ps	100 ps	200 ps			500 ps

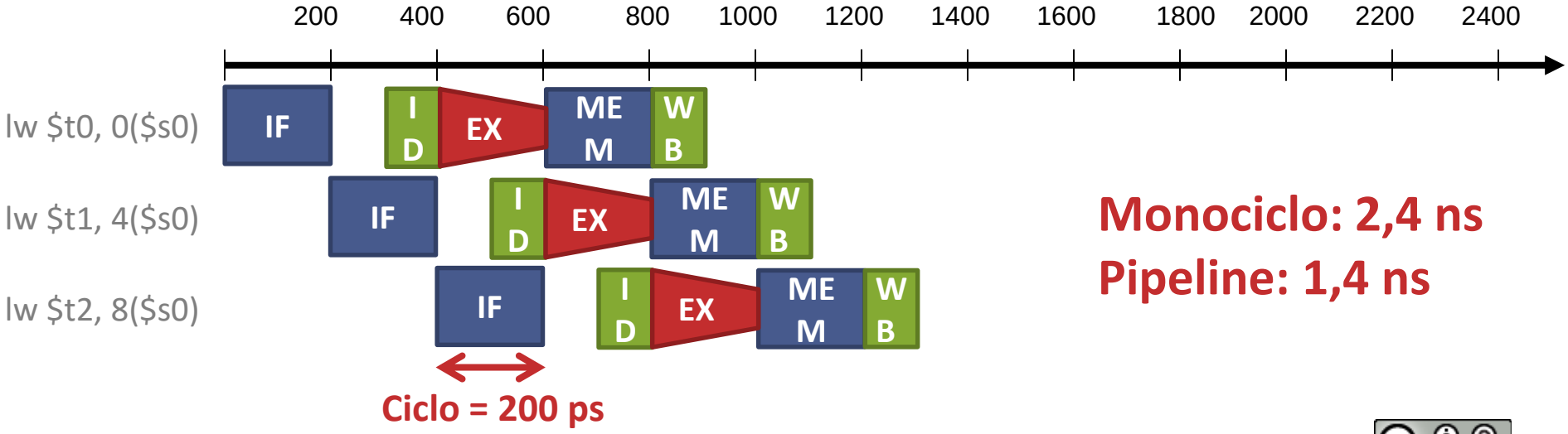
  - Tempo de ciclo
    - **Monociclo: 800 ps** (lw como pior caso)
    - **Pipeline: 200 ps** (memória ou ULA como pior caso)

Pipelining

- Monociclo



- Pipeline



Monociclo: 2,4 ns
Pipeline: 1,4 ns

Pipelining

- **Aceleração ideal = número de estágios**
  - Maior número de ciclos para uma instrução compensado por ciclo mais curto
  - Depois do pipeline encher: um ciclo, uma instrução
    - CPI próximo de 1

Pipelining

- **Exemplo de cálculo de desempenho**
  - Dado um processador pipeline de 8 estágios, informe (i) **quantos ciclos** ele levará para executar as quantidades de instruções abaixo e (ii) os **CPIs médios** para cada caso:
    - 2 instruções
    - 10 instruções
    - 100 instruções
    - 10000 instruções

Pipelining

- Pipelining é fácil
  - ~10% a mais de hardware, muito mais desempenho
  - **Pipelining não é fácil!**

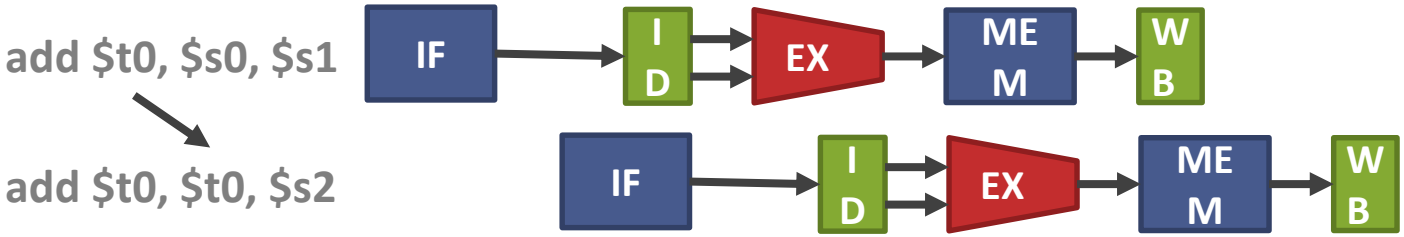
HAZARDS

Hazards

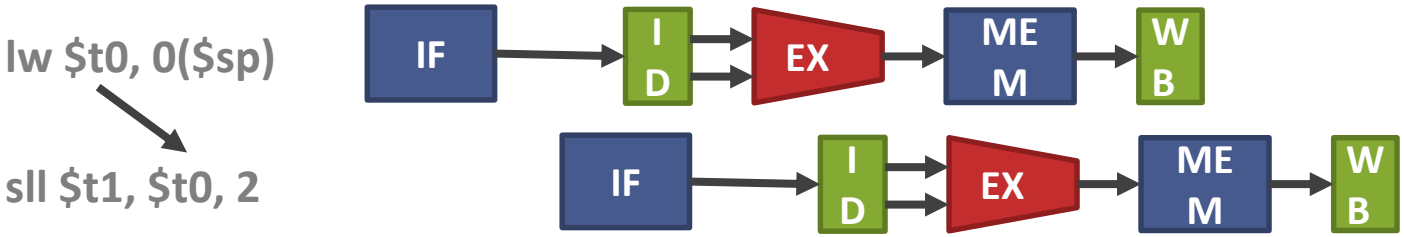
- Problemas de pipelines: hazards
  - **Hazards estruturais**
    - Duas instruções precisando dos mesmos recursos de hardware ao mesmo tempo
  - **Hazards de dados**
    - Dados necessários para uma instrução ainda não estão disponíveis
  - **Hazards de controle (ou de desvio)**
    - Instrução buscada não é a que deve ser executada

Hazards

- Exemplos de hazard de dados
  - add $t0, $s0, $s1 seguido de add $t0, $t0, $s2

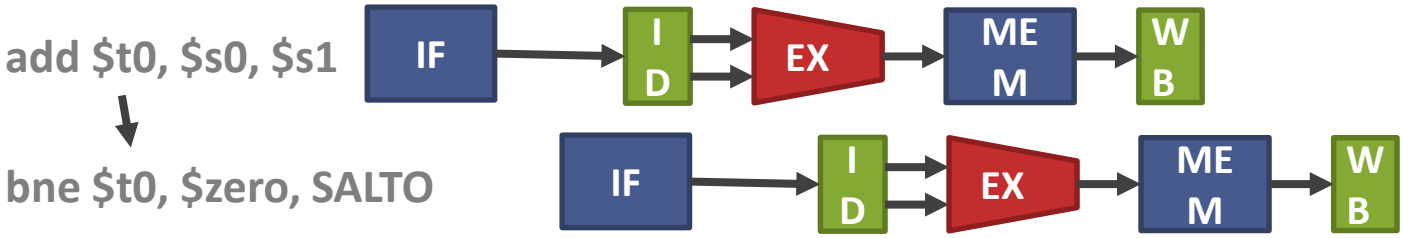


  - lw $t0, 0($sp) seguido de sll $t1, $t0, 2



Hazards

- Exemplos de hazard de controle
  - add $t0, $s0, $s1 seguido de bne usando $t0



  - Qual é a próxima instrução?
    - Instrução após bne?
    - Instrução no endereço do label SALTO?

Hazards

- Soluções para hazards gerais
  - **Reordenação de instruções no código**
    - Instruções independentes em sequência
      - Nem sempre é possível
  - ***Stalls***
    - Inserção de bolhas no pipeline quando instruções não podem ser executadas
      - Pode atrasar muito a execução do código

Hazards

- Exercício
  - Dado um processador MIPS com pipeline de cinco estágios e um código onde 20% das instruções levam a hazards que necessitam de *stalls* de um ciclo no pipeline, qual é o CPI aproximado obtido?
  - Qual o CPI para o caso de serem executadas 10 instruções?

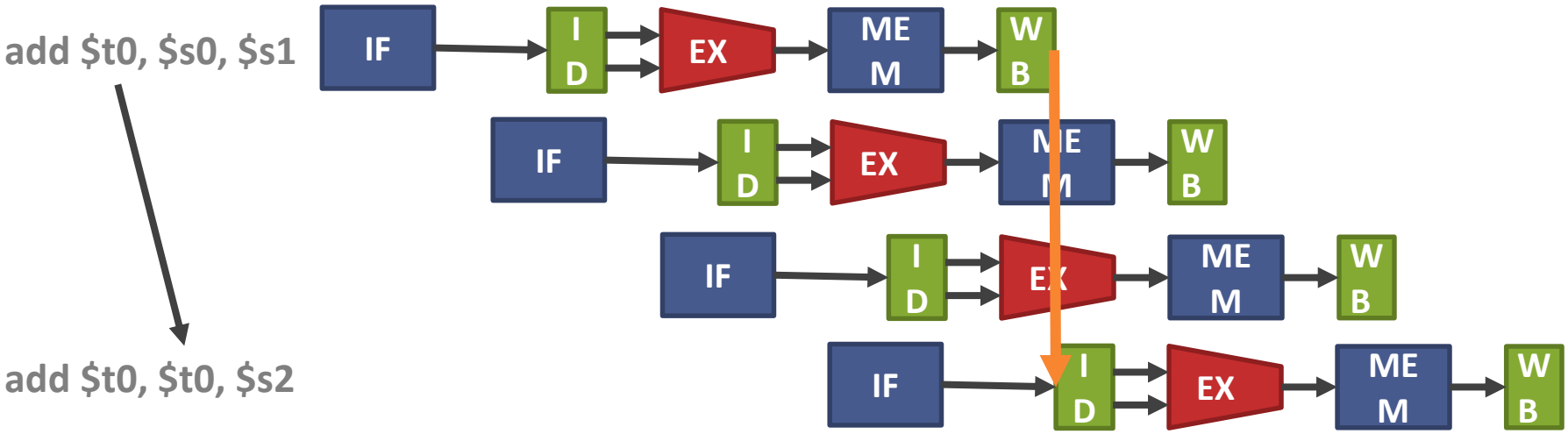
DATA HAZARDS E FORWARDING

Data hazards e forwarding

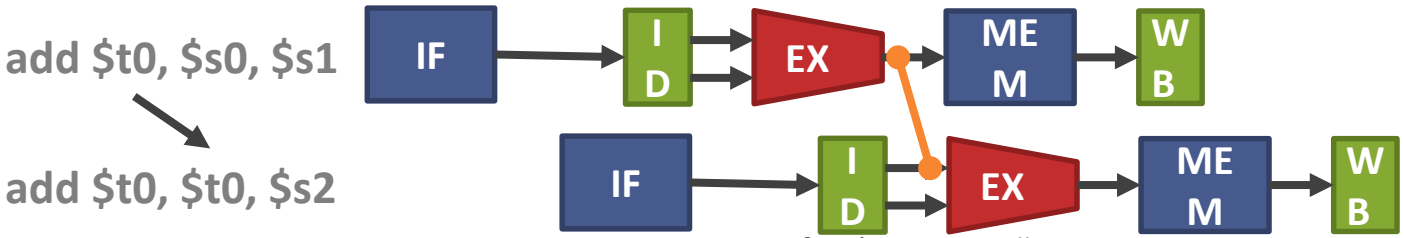
- Soluções para hazard de dados
  - ***Forwarding***
    - Encaminhamento de valores já calculados para serem utilizados por instruções antes da escrita no banco de registradores

Data hazards e forwarding

- Exemplo
  - add $t0, $s0, $s1 seguido de add $t0, $t0, $s2
  - Sem *forwarding*

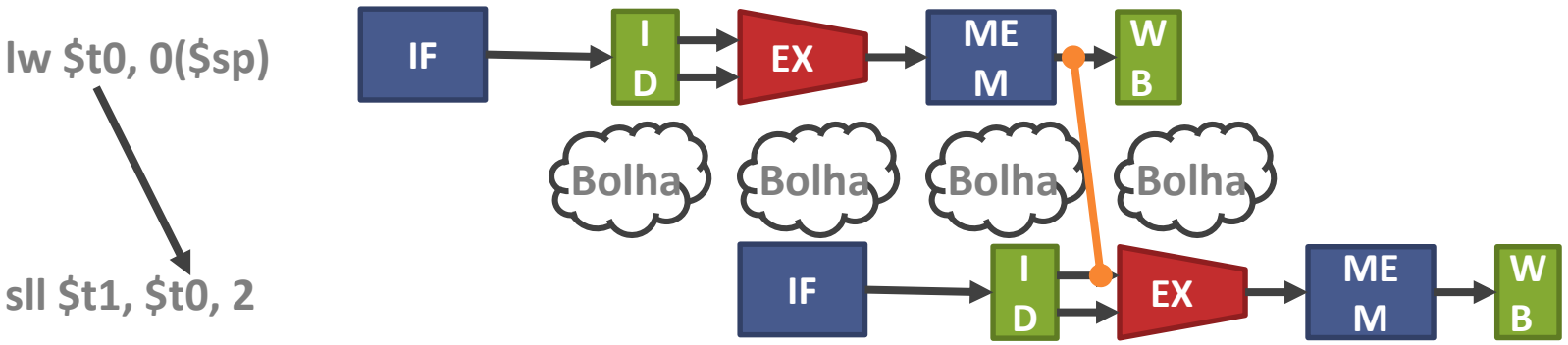


  - Com *forwarding*



Data hazards e forwarding

- *Forwarding + stalls*
  - lw $t0, 0($sp) seguido de sll $t1, $t0, 2



  - Quantos ciclos levaria para termos os dados sem *forwarding*?

Data hazards e forwarding

- Exemplo
  - Detecte as dependências entre as instruções abaixo
    - andi $t0, $s0, 0x00FF
    - or $t0, $t0, $s1
    - or $s0, $s0, $s1
    - sw $t0, 0($s5)
  - Informe quantos ciclos são necessários para a execução do código apenas com bolhas (*stalls*) e com *forwarding*

CONTROL HAZARDS E PREDIÇÃO DE DESVIOS

Control hazards e predição de desvios

- Exemplo de hazard de controle

```
beq $t0, $zero, SAIDA
addi $t0, $t0, 1
SAIDA:
sub $t2, $t1, $t0
```

  - Qual instrução deve executar após o branch?

Control hazards e predição de desvios

- Soluções para hazards de controle

"Quartz crystal" by Sanjay Acharya - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Quartz_crystal.jpg#/media/File:Quartz_crystal.jpg

Control hazards e predição de desvios

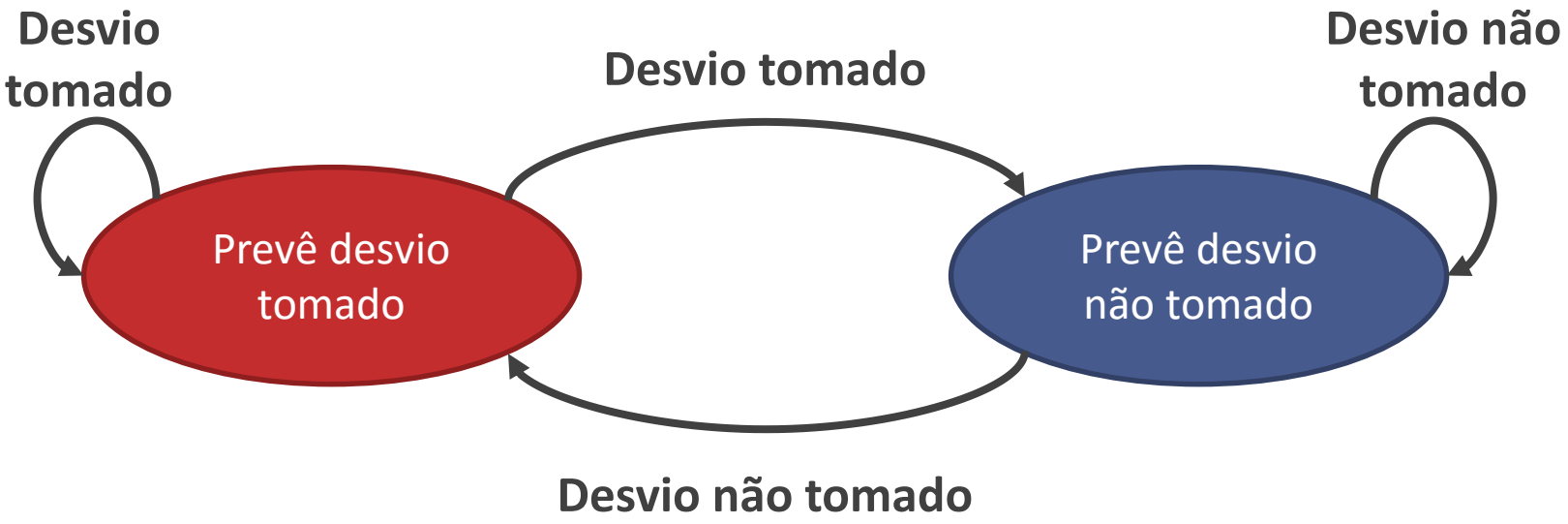
- Soluções para hazards de controle
  - ***Delay slot***
    - Coloca instruções após o branch para serem executadas de qualquer forma
  - **Execução especulativa**
    - Exemplo: assume desvio não tomado
    - Executa a instrução logo após o branch
    - Só é finalizada se o desvio não for tomado
  - **Previsão de desvios**
    - Mecanismos para tentar adivinhar se o desvio será tomado ou não

Control hazards e predição de desvios

- **Previsão de desvios dinâmica**
  - Uso de um histórico dos desvios para a tomada de decisões
    - Tabela de histórico de desvios (BHT)
  - Exemplo
    - Um desvio foi tomado recentemente? Provavelmente vai acontecer novamente
    - O desvio desta instrução nunca é tomado, então vamos executar a próxima instrução
  - No caso de previsão errada, joga as instruções especuladas fora

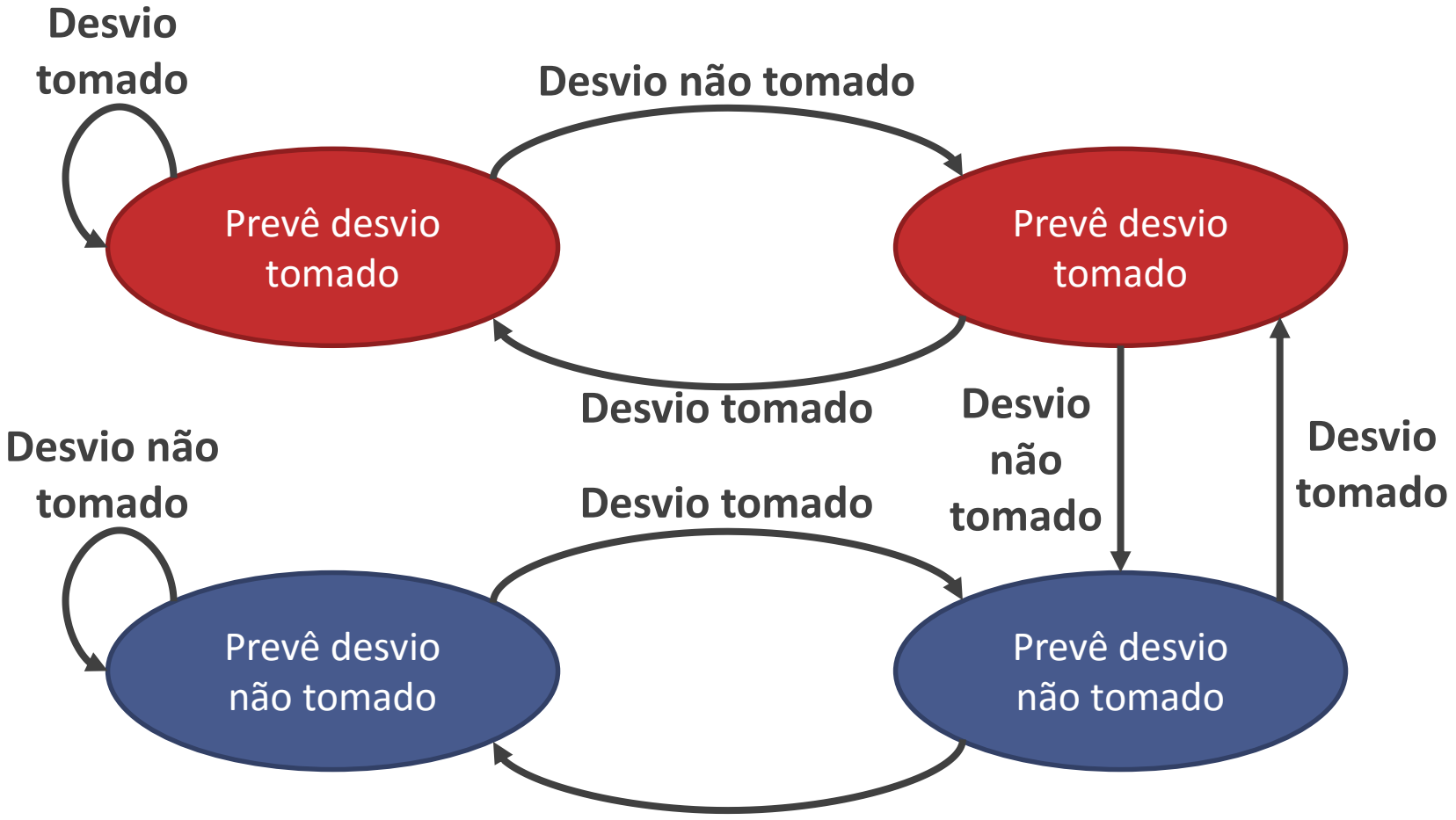
Control hazards e predição de desvios

- **Previsão de desvios dinâmica**
  - Previsor de um bit



Control hazards e predição de desvios

- **Previsão de desvios dinâmica**
  - Previsor de dois bits



CONSIDERAÇÕES FINAIS

Considerações finais

- **Pipelining**
  - Técnica usada em praticamente todos os processadores atuais
  - Exige mecanismos mais elaborados para seu uso eficiente
    - *Forwarding*
    - Previsão de desvios

Considerações finais

- Pipeline não é só hardware
  - Exemplos de pipeline em software
    - Aplicação de filtros em sequência
    - Aplicação de comandos com pipe “|”
    - Decodificação de mp3
    - Criptografia
    - etc.

Considerações finais

- Exemplos de processadores pipeline na próxima aula!

INE5607 – Organização e Arquitetura de Computadores

Unidade Central de Processamento

Aula 16: Processadores pipeline

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br