Universidade Federal de Pelotas
Instituto de Física e Matemática
Departamento de Informática
Bacharelado em Ciência da Computação

Arquitetura e Organização de Computadores II

Aula 8

2. MIPS pipeline: visão geral do pipeline, conflitos na execução em pipeline.

Prof. José Luís Güntzel

guntzel@ufpel.edu.br

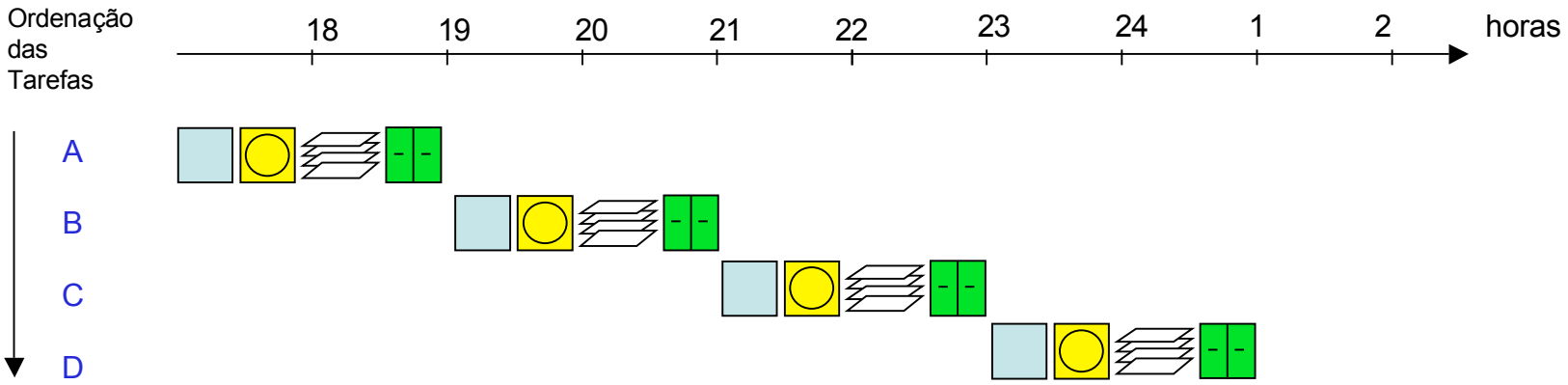
www.ufpel.edu.br/~guntzel/AOC2/AOC2.html

2. Organizações do MIPS: pipeline

Visão Geral do Pipeline

Analogia com uma "Lavanderia Doméstica" 1:

- **Execução seqüencial**



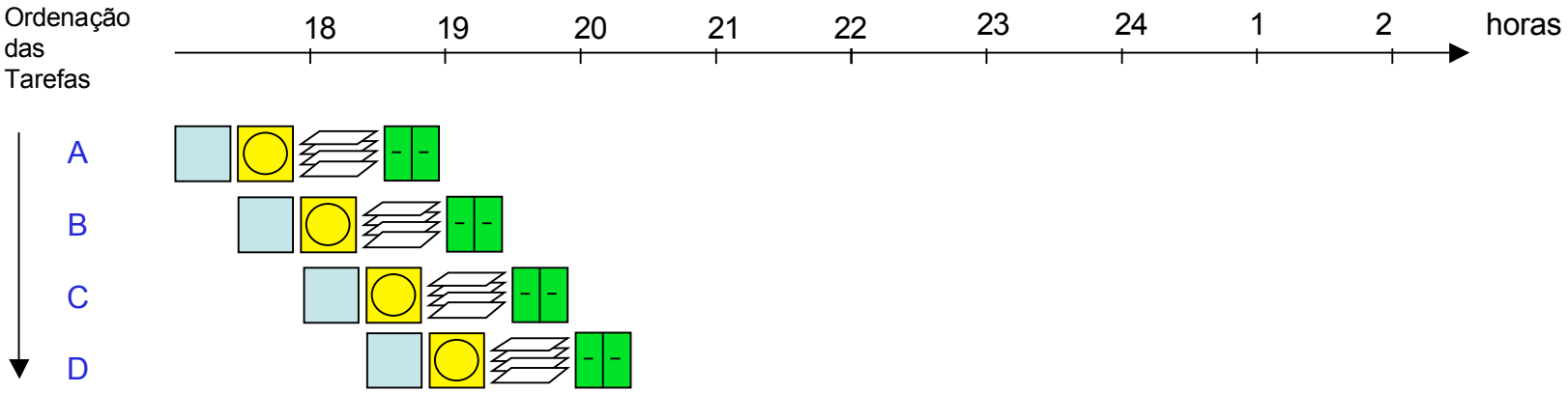
Tempo total: 8 horas
Tempo entre duas mudas de roupas prontas: 2 horas

2. Organizações do MIPS: pipeline

Visão Geral do Pipeline

Analogia com uma "Lavanderia Doméstica" 2:

- Execução em pipeline



Tempo total: 3,5 horas
Tempo entre duas mudas de roupas prontas: 1/2 hora

2. Organizações do MIPS: pipeline

Visão Geral do Pipeline

Tradicionalmente, as instruções do MIPS são executadas em 5 etapas:

1. **Busca da instrução na memória**
2. **Leitura dos registradores, enquanto a instrução é decodificada**
3. **Execução de uma operação ou cálculo de um endereço**
4. **Acesso a um operando na memória**
5. **Escrita do resultado em um registrador**

2. Organizações do MIPS: pipeline

Visão Geral do Pipeline

Exemplo 1: compare o tempo de execução da versão monociclo do MIPS com o tempo da versão pipeline.

Considere os tempos de operação para as principais unidades funcionais (e os tempos das instruções) dados na tabela abaixo:

Classe da instrução	Busca da instrução	Leitura de registrador	Operação na ULA	Acesso ao dado	Escrita no registrador	Tempo total
Load word (~~ld~~) lw	2 ns	1 ns	2 ns	2 ns	1 ns	8 ns
Store word (sw)	2 ns	1 ns	2 ns	2 ns	---	7 ns
Formato R (add, sub, and, or, slt)	2 ns	1 ns	2 ns	---	1 ns	6 ns
Branch (beq)	2 ns	1 ns	2 ns	---	---	5 ns

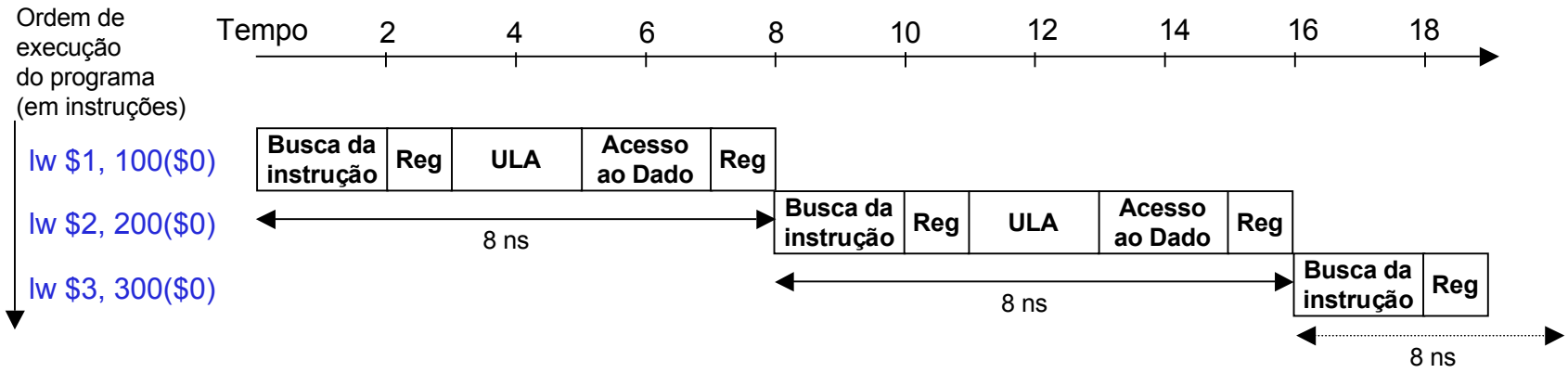
2. Organizações do MIPS: pipeline

Visão Geral do Pipeline

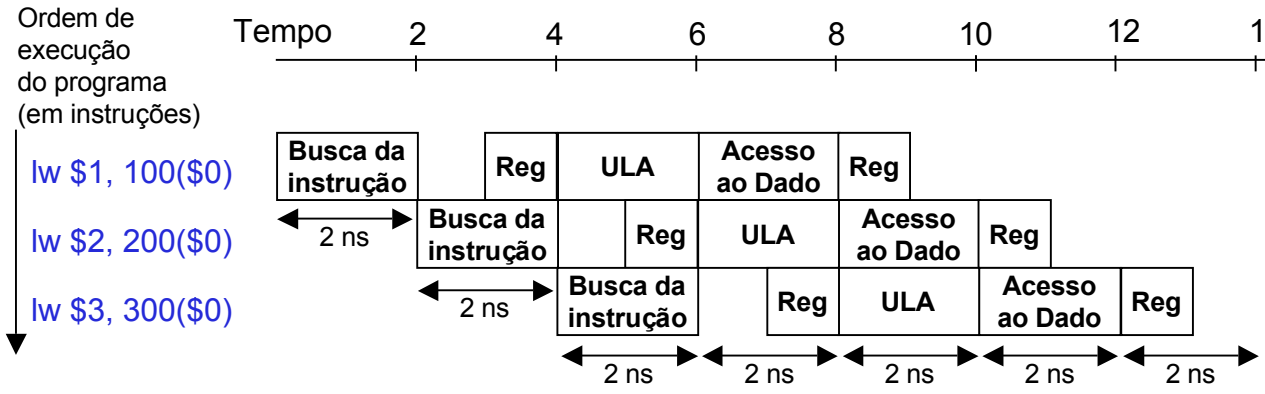
Consideremos 3 execuções seguidas da instrução `lw`, a qual é a mais lenta em ambos casos (monociclo e pipeline).

2. Organizações do MIPS: pipeline

sem pipeline



com pipeline



Ganho do pipeline = 24/6 = 4

2. Organizações do MIPS: pipeline

Visão Geral do Pipeline

- **Sem pipeline: o tempo decorrido entre o início da primeira instrução e o início da quarta instrução é 3x8 ns = 24 ns**
- **Com pipeline: o tempo decorrido entre o início da primeira instrução e o início da quarta instrução é 3x2 ns = 6 ns**
- **Logo, a melhora do desempenho advinda do uso do pipeline é de 24 ns/6 ns = 4 vezes!**

2. Organizações do MIPS: pipeline

Visão Geral do Pipeline

Se os estágios de um pipeline forem perfeitamente balanceados, então:

$$\text{Tempo entre instruções}_{\text{pipeline}} = \frac{\text{Tempo entre instruções}_{\text{não-pipeline}}}{\text{Número de estágios do pipe}}$$

- **Estas condições são ideais e portanto, nunca atingidas!**
- **Porém, costuma-se assumir que o ganho máximo teórico de um pipeline é igual ao número de estágios.**

2. Organizações do MIPS: pipeline

Visão Geral do Pipeline

Motivos pelos quais o ganho teórico não é atingido:

- **Os estágios de um pipeline não são perfeitamente balanceados**
- **Existe um *overhead* referente ao preenchimento do pipeline.**

Em função do segundo motivo, o tempo entre instruções é mais importante do que o tempo total.

2. Organizações do MIPS: pipeline

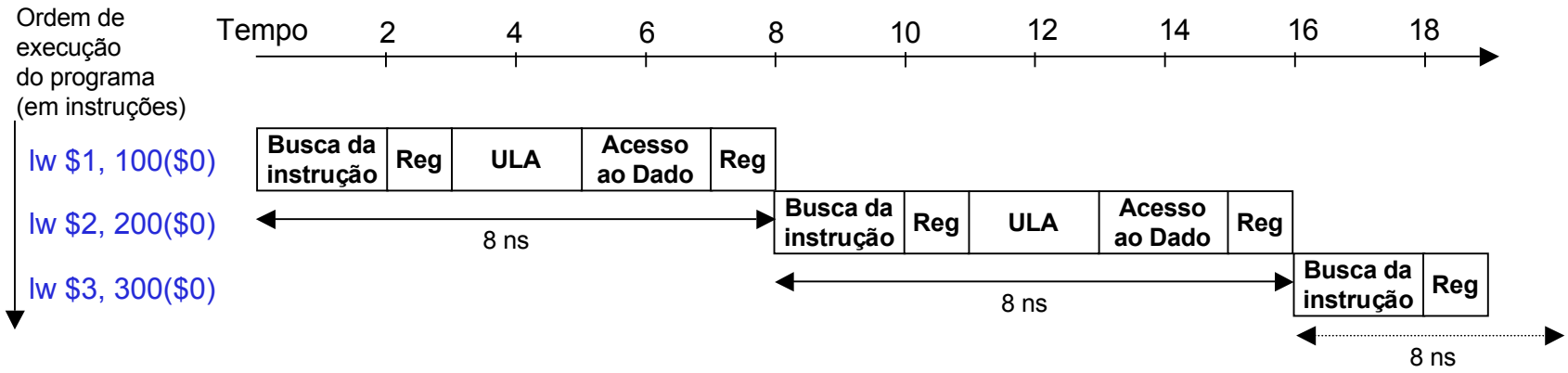
Visão Geral do Pipeline

Exemplo 2: ainda com relação ao exemplo 1, considere que ao invés de 3 instruções `lw`, sejam executadas 1003 instruções `lw` seguidas.

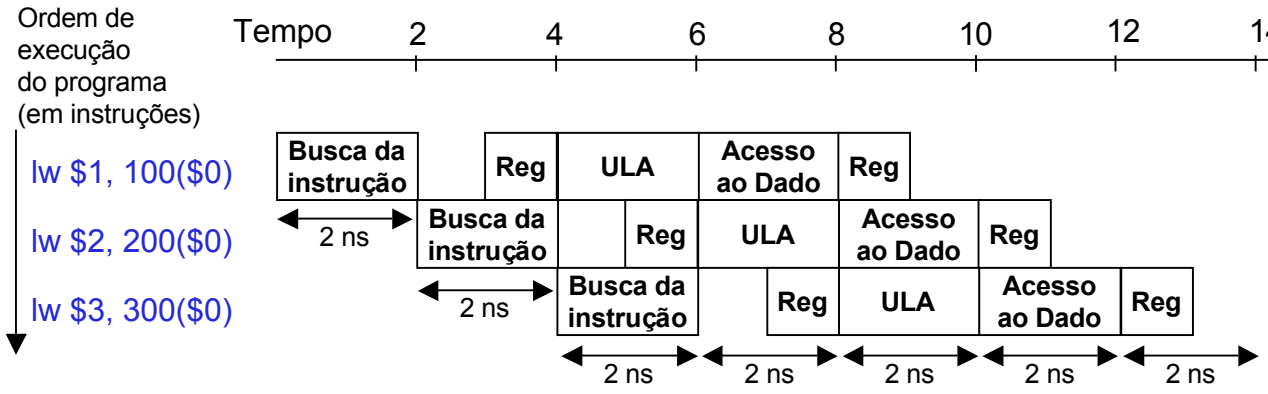
Qual será o tempo total em cada caso (monociclo e pipeline) e qual será o ganho obtido pelo uso do pipeline?

2. Organizações do MIPS: pipeline

sem pipeline



com pipeline



2. Organizações do MIPS: pipeline

Visão Geral do Pipeline

- **Sem pipeline:** 24 ns + 1000 x 8 ns = 8024 ns
- **Com pipeline:** ~~14~~ (6 ns) ns + 1000 x 2 ns = 2014 ns

Ganho = 8024 ns / ~~2014~~ (2006) ns = ~~3,98~~ (4) ≈ 8 ns / 2 ns

O pipeline aumenta o desempenho por meio do aumento do *throughput* das instruções, ou seja, aumentando o número de instruções executadas na unidade de tempo (e não por meio da diminuição do tempo de execução de uma instrução individual).

2. Organizações do MIPS: pipeline

Projeto da Arquitetura para o MIPS Pipeline (Conjunto de Instruções)

Características da Arquitetura do Processador MIPS*:

1. **Todas as instruções têm o mesmo tamanho**
2. **Poucos formatos de instruções, sempre com o registrador-fonte localizado na mesma posição**
3. **Somente as instruções load word e store word manipulam operandos na memória**
4. **Os operandos do MIPS precisam estar alinhados na memória**

* **Na realidade, o conjunto de instruções do MIPS foi projetado visando a execução em pipeline.**

2. Organizações do MIPS: pipeline

Os Conflitos do Pipeline

"Existem situações de execução no pipeline em que a instrução seguinte não pode ser executada no próximo ciclo de relógio. Tais situações são chamadas de conflitos."

Tipos de Conflitos:

- **Estruturais**
- **De controle**
- **De dados**

2. Organizações do MIPS: pipeline

Conflitos Estruturais (*Structural Hazards*)

O Hardware não pode suportar a combinação de instruções que o pipeline deseja executar em um dado ciclo de relógio.

- O conjunto de instruções do MIPS foi projetado para executar em pipeline (é muito fácil evitar conflitos estruturais quando do desenho do pipeline)
- Mas se houvesse somente uma memória (para dados e instruções), se uma instrução tentasse acessar um dado na memória enquanto tivesse que ser buscada uma nova instrução, ocorreria um conflito estrutural.

2. Organizações do MIPS: pipeline

Conflitos de Controle (*Control Hazards*)

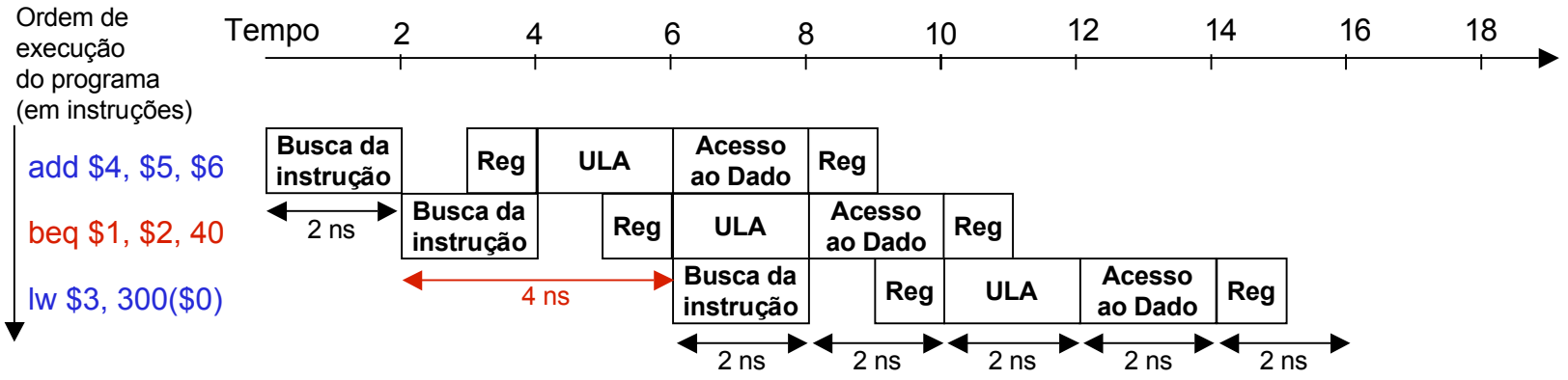
- **Originam-se na necessidade de se tomar uma decisão baseada nos resultados de uma instrução, a qual ainda não foi concluída.**
- **Muito comuns em instruções de salto condicional (`beq`, no caso do MIPs reduzido)**
- **Uma possível solução é a parada (também chamada de "bolha"), ou seja, interromper a progressão das instruções pelo pipeline.**

2. Organizações do MIPS: pipeline

Conflitos de Controle: parada

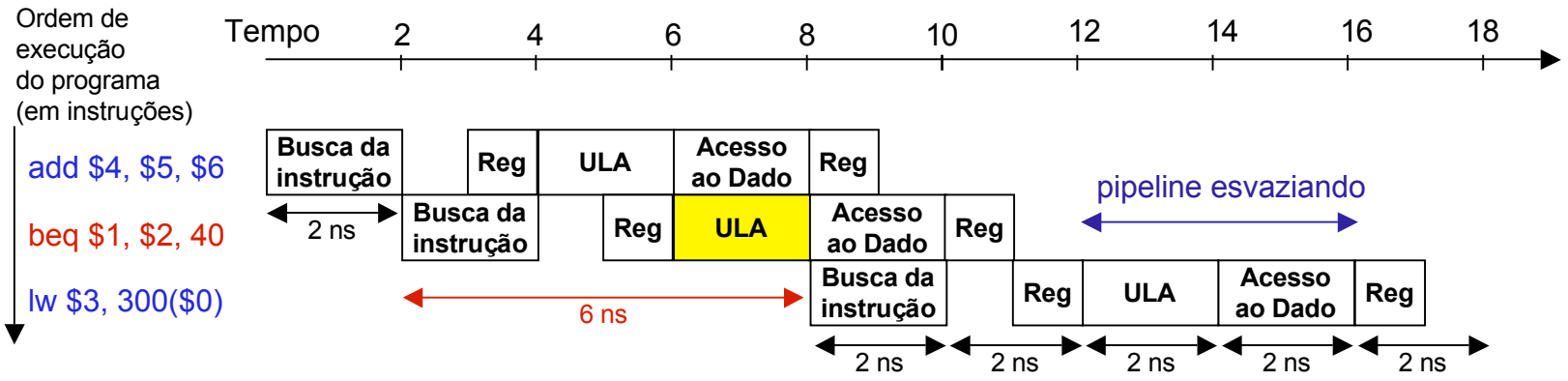
Exemplo 3: analisar o efeito da parada no desempenho do desvio condicional.

Considere que se tenha hardware extra que permita testar registradores, calcular o endereço de desvio condicional e atualizar o PC, tudo isso no segundo estágio.



2. Organizações do MIPS: pipeline

Conflitos de Controle: parada



Se for necessário resolver o desvio condicional em um estágio posterior ao segundo (o que é o ocorre na maioria dos processadores reais), ocorrerá uma perda maior de desempenho em função da necessidade de encher novamente o pipeline

2. Organizações do MIPS: pipeline

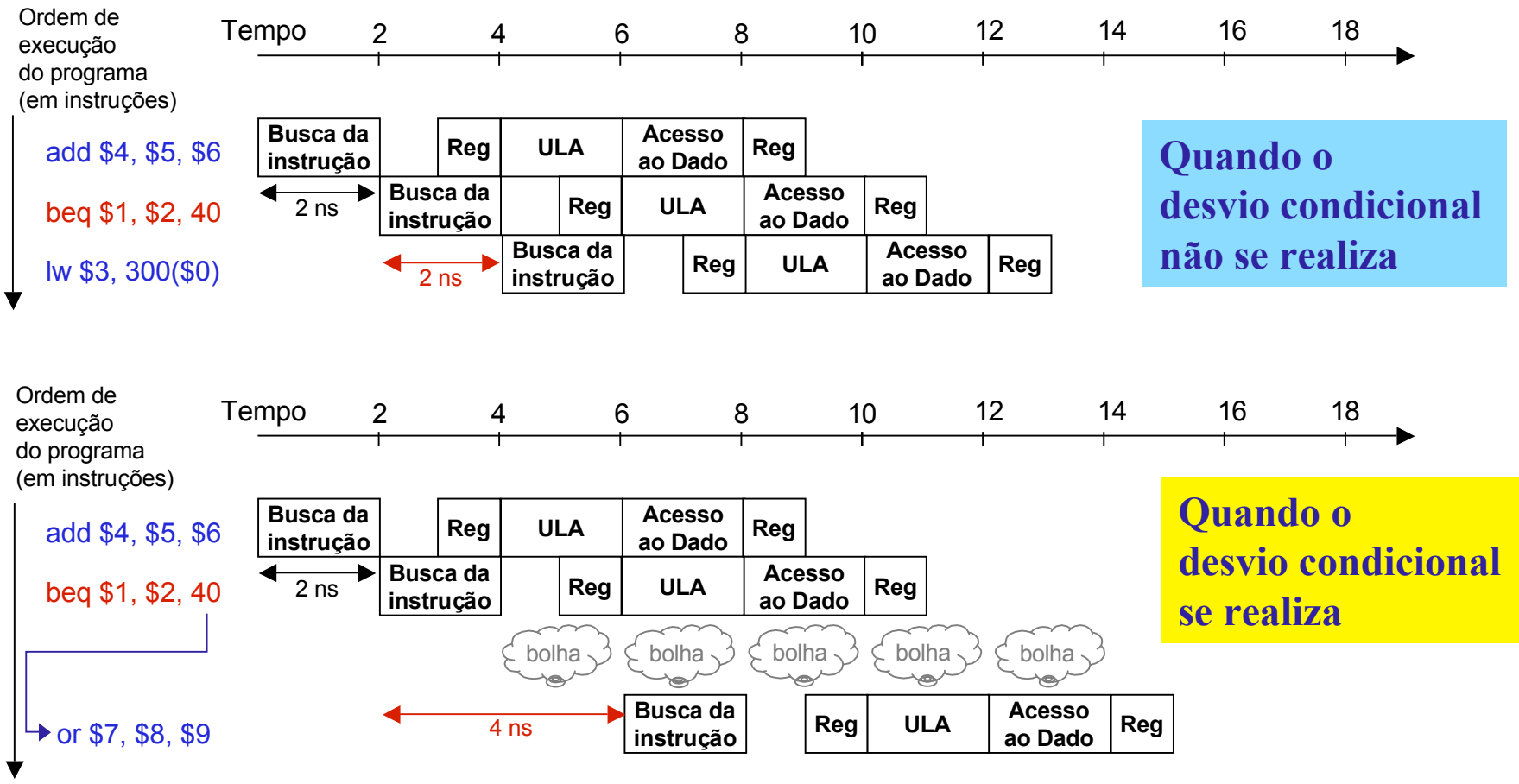
Conflitos de Controle: predição estática

Outra solução é a Predição:

- **É a técnica geralmente adotada nos processadores reais!**
- **Um esquema simples de predição é assumir sempre que os desvios condicionais sempre irão falhar.**
  - **Se acertar, o pipeline prossegue com velocidade máxima**
  - **Se errar, será neccesário atrasar o avanço das instruções pelo pipeline**

2. Organizações do MIPS: pipeline

Conflitos de Controle: predição estática



2. Organizações do MIPS: pipeline

Conflitos de Controle: predição estática

- **Outro esquema mais sofisticado de predição reside em se considerar parte dos desvios se realizando e parte não se realizando**
- **Exemplo de uso prático: nos loops a última instrução é sempre um desvio para um endereço anterior, que na maioria das vezes, se concretiza**
- **Dada a grande probabilidade de se realizarem, podemos predizer que os desvios para endereços anteriores sempre se realizam…**

2. Organizações do MIPS: pipeline

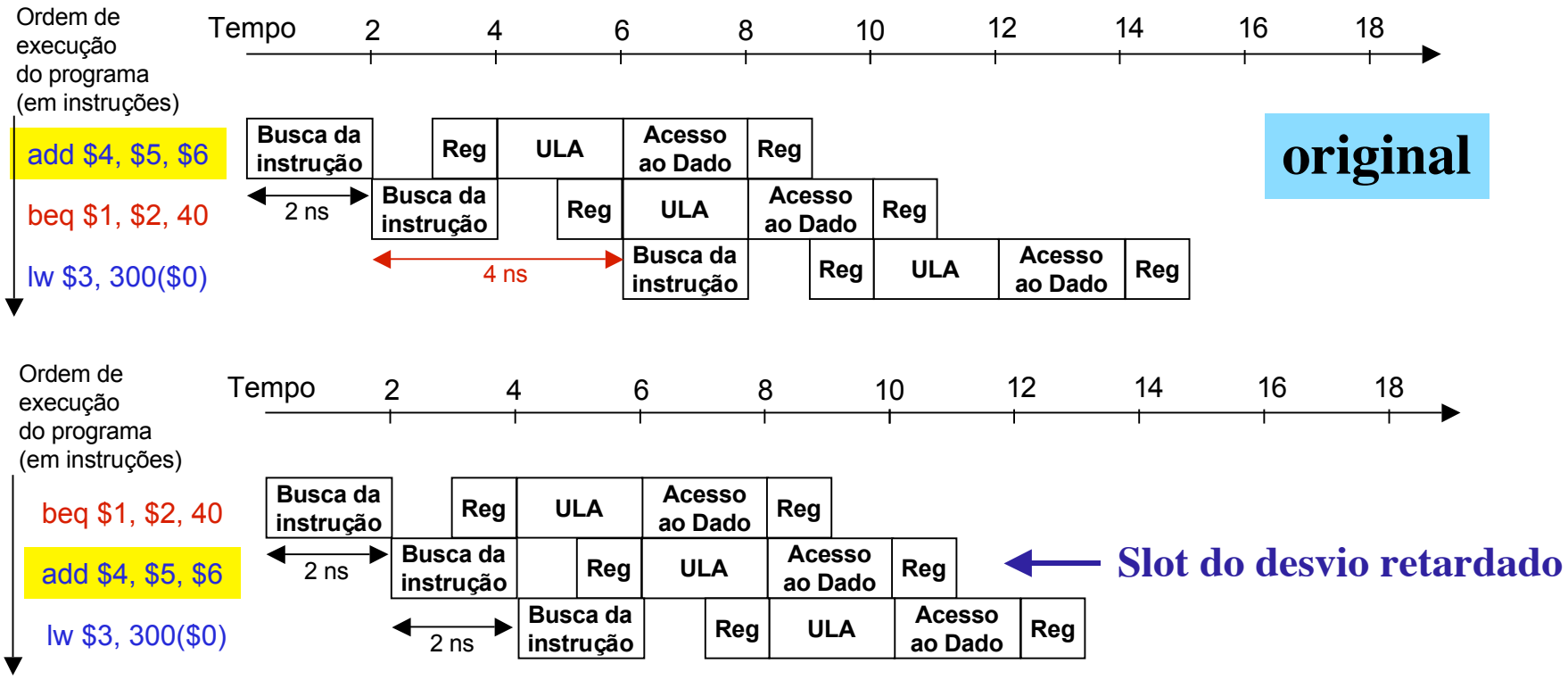
Conflitos de Controle: predição dinâmica

- **Predição dinâmica realiza a predição em função do comportamento anterior de cada desvio**
- **Podem mudar a predição para um mesmo desvio com o decorrer da execução do programa**
- **Um esquema comum é manter uma história para cada desvio realizado ou não-realizado**
- **Preditores dinâmicos são implementados em hardware e apresentam até 90% de acerto**
- **Quando o palpite estiver errado, o controle do pipeline precisa assegurar que as instruções seguintes ao desvio não terão influência na execução futura**

2. Organizações do MIPS: pipeline

Conflitos de Controle: *delayed branch*

- **Decisão retardada** **consiste em escolher uma instrução para ser executada logo após a instrução de desvio, de modo a manter o pipeline preenchido**

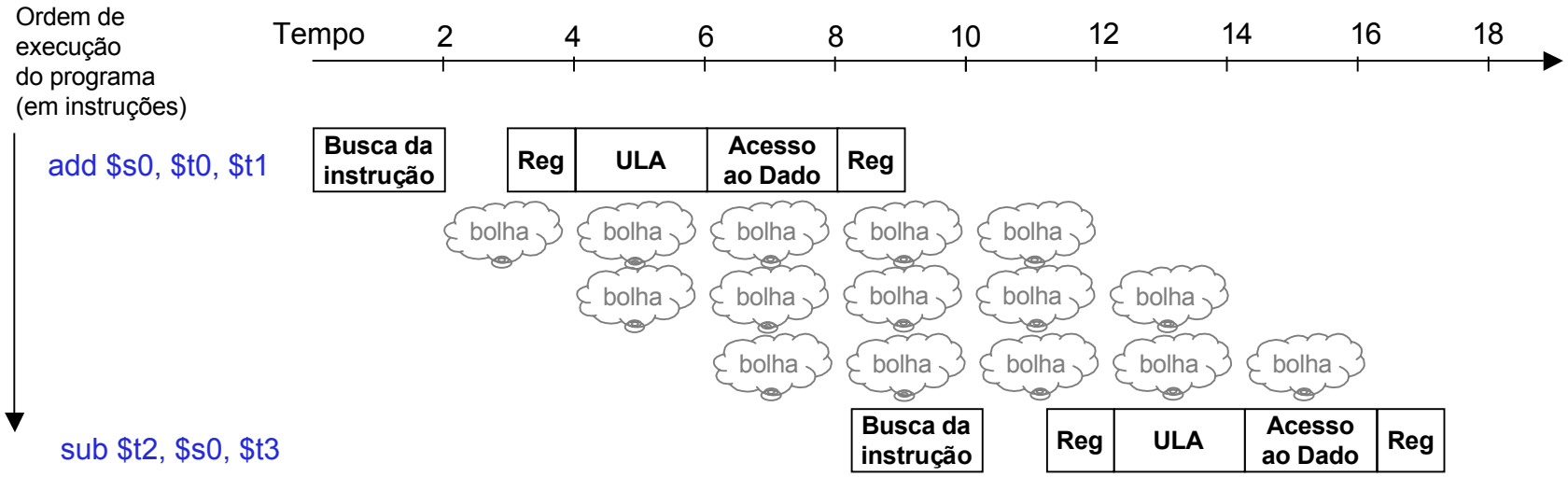


2. Organizações do MIPS: pipeline

Conflitos de Dados (*Data Hazards*)

- **A execução de uma instrução depende do resultado de outra, a qual ainda está no pipeline. Exemplo:**

```
add   $s0, $t0, $t1
sub   $t2, $s0, $t3
```



2. Organizações do MIPS: pipeline

Conflitos de Dados: adiantamento

- **Passar para o compilador a solução deste tipo de conflito é inviável, pois este tipo de conflito é muito freqüente**
- **Solução: não é preciso esperar pelo término de uma instrução aritmética/lógica**
- **Tão logo a ULA chegue ao resultado da operação, este resultado pode ser disponibilizada para as instruções que vem a seguir**
- **Esta técnica é chamada de adiantamento (*forwarding* ou *bypass*)**

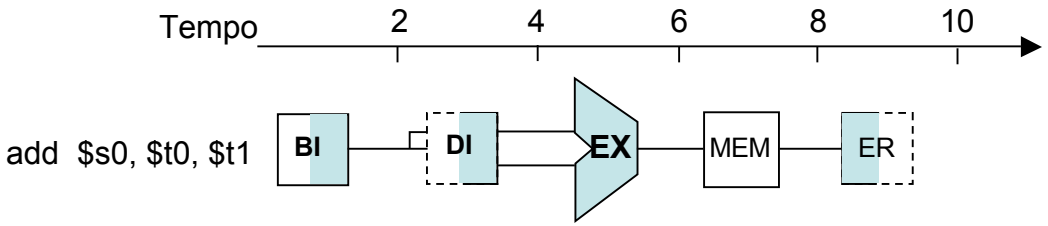
2. Organizações do MIPS: pipeline

Conflitos de Dados: adiantamento

Exemplo 4: considerando as duas instruções citadas anteriormente, mostre as conexões necessárias entre os estágios do pipeline de modo a tornar viável o adiantamento.

2. Organizações do MIPS: pipeline

Representação Gráfica do Pipeline de Instrução



Estágios:

BI: busca de instrução (memória de instruções)

DI: decodificação da instrução e leitura do banco de registradores (banco de registradores sendo lido)

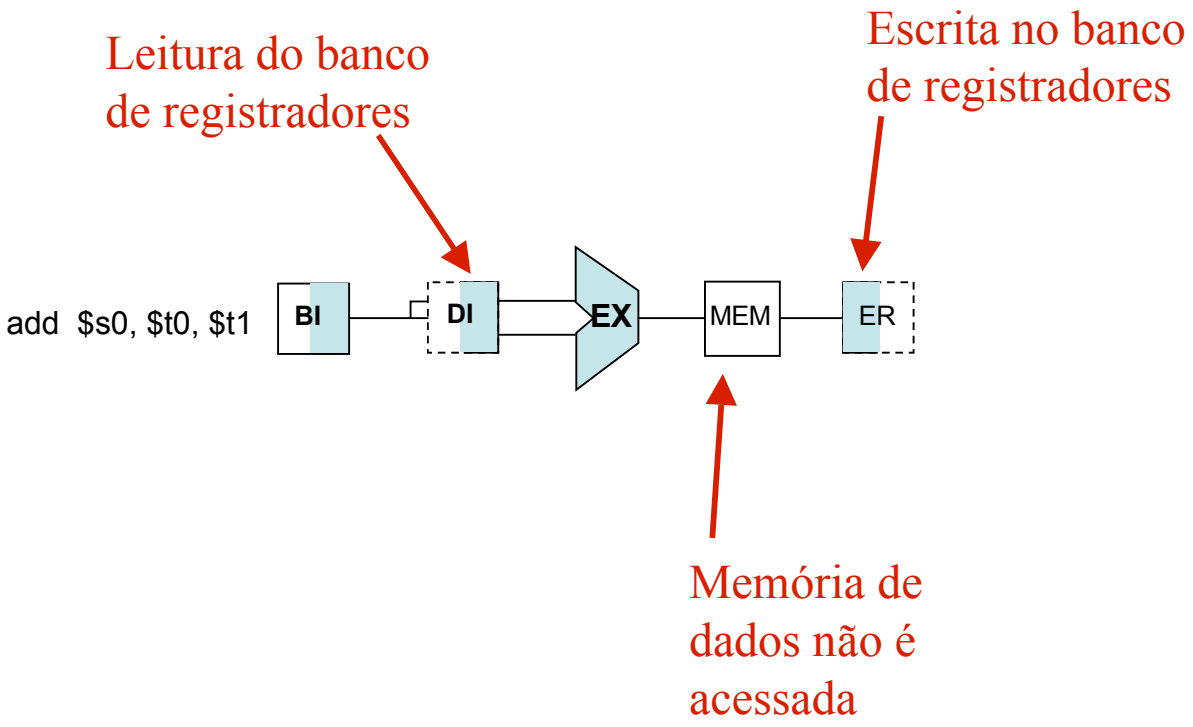
EX: estágio de execução da instrução (ULA)

MEM: acesso à memória (memória de dados)

ER: escrita do resultado no banco de registradores (banco de registradores sendo escrito)

2. Organizações do MIPS: pipeline

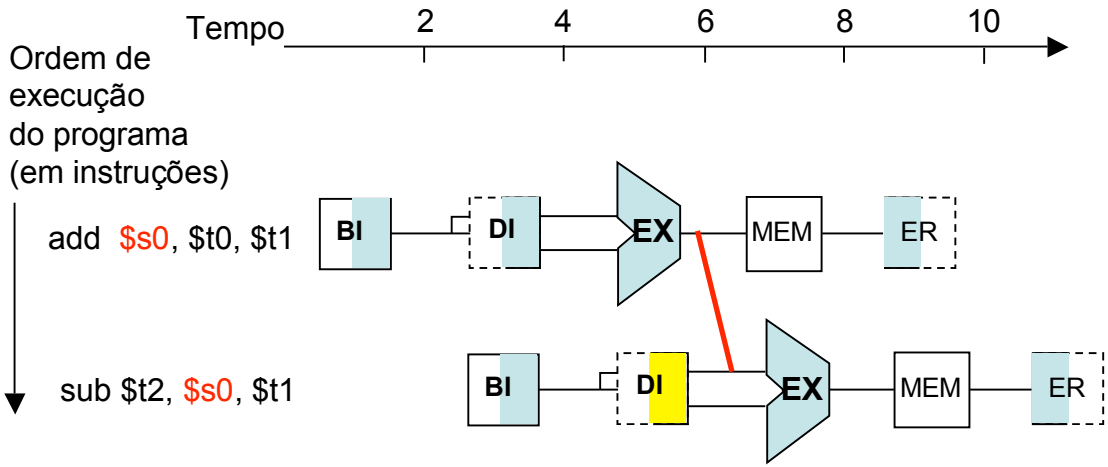
Representação Gráfica do Pipeline de Instrução



2. Organizações do MIPS: pipeline

Conflitos de Dados: adiantamento

Solução do Exemplo 4: conexão entre a saída da ULA e a entrada da própria ULA



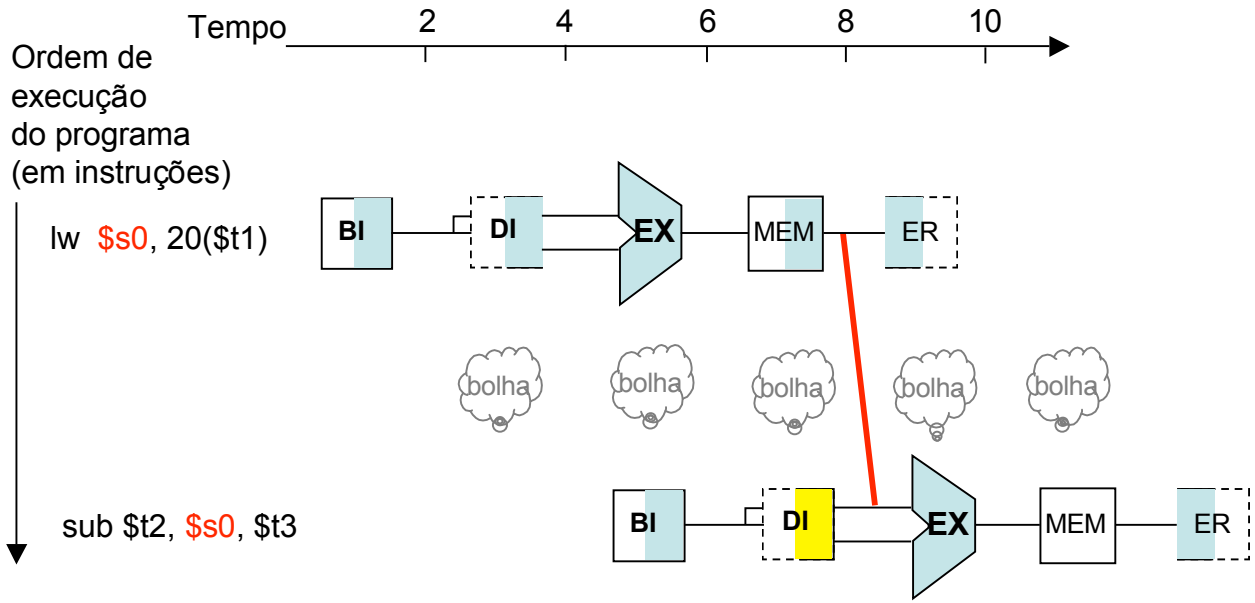
Resultado: não haverá bolha!

Só pode haver caminho para adiantamento se o estágio-destino for posterior no tempo ao estágio-fonte

2. Organizações do MIPS: pipeline

Conflitos de Dados: adiantamento

Exemplo 5: adiantamento envolvendo instrução `lw`



Resultado: mesmo com adiantamento, haverá a perda de um ciclo de relógio (a bolha inevitável)

2. Organizações do MIPS: pipeline

Conflitos de Dados: reordenação do código

Exemplo 6: encontre o conflito existente no código do procedimento `swap`, abaixo.

```
                    # reg $t1 possui o endereco de v[k]
lw $t0, 0($t1)      # reg $t0 (temp) = v[k]
lw $t2, 4($t1)      # reg $t2 = v[k+1]
sw $t2, 0($t1)      # v[k] = reg $t2
sw $t0, 4($t1)      # v[k+1] = reg $t0 (temp)
```

O conflito aparece no registrador `$t2`, entre a 2ª e a 3ª linhas.

2. Organizações do MIPS: pipeline

Conflitos de Dados: reordenação do código

O conflito aparece no registrador `$t2`, entre a 2ª e a 3ª linhas.

Solução: intercambiar as duas últimas instruções de `sw`.

```
                    # reg $t1 possui o endereco de v[k]
lw $t0, 0($t1)      # reg $t0 (temp) = v[k]
lw $t2, 4($t1)      # reg $t2 = v[k+1]
sw $t0, 4($t1)      # v[k+1] = reg $t0 (temp)
sw $t2, 0($t1)      # v[k] = reg $t2
```

Note que nenhum novo conflito foi criado!