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

Arquitetura e Organização de Computadores II

Aula 10

2. MIPS pipeline: conflitos por dados e adiantamento.

Prof. José Luís Güntzel

guntzel@ufpel.edu.br

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

2. Organizações do MIPS: pipeline

Conflitos por Dados

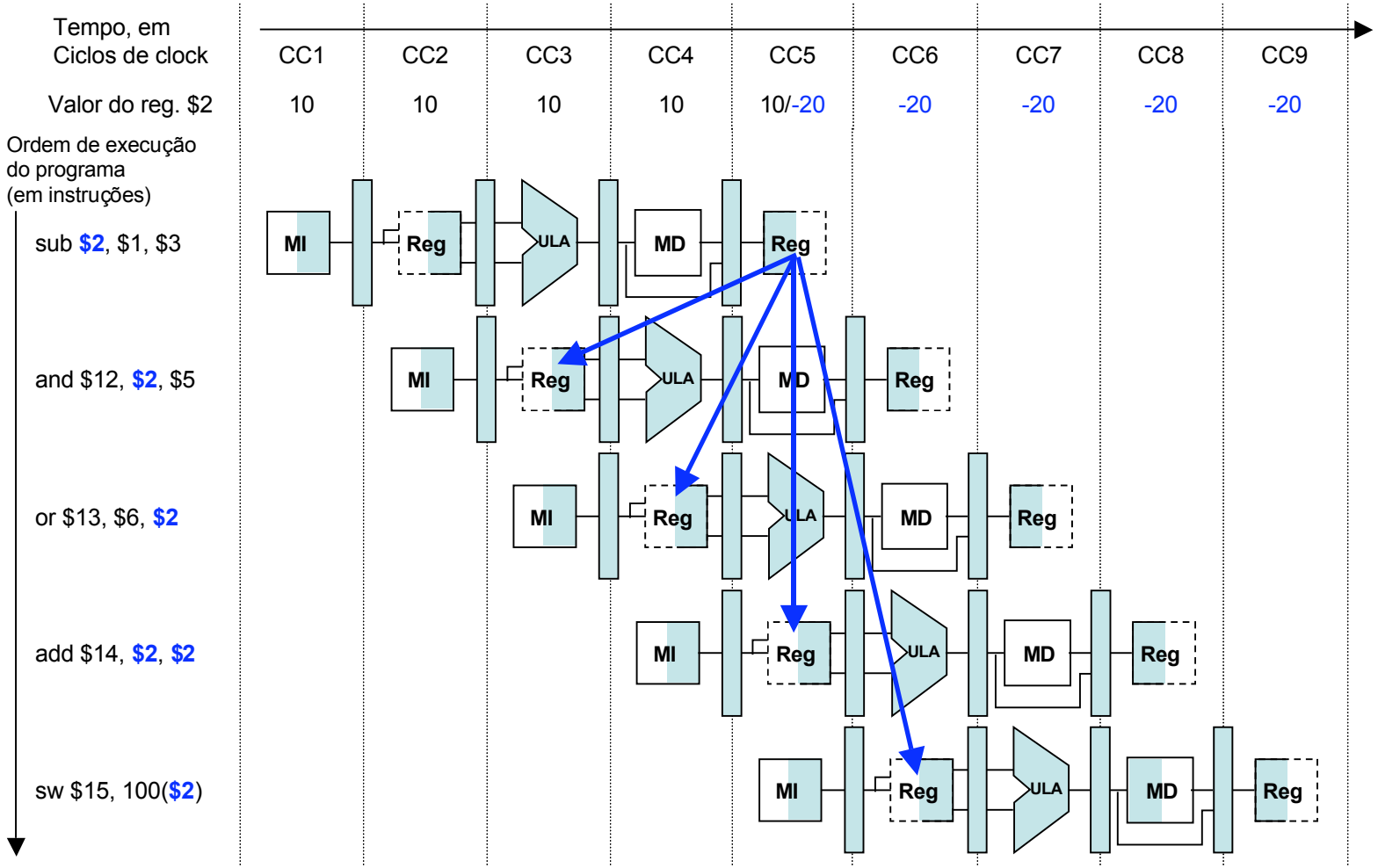
Seja o seguite trecho de código, escrito para o MIPS

```
sub   $2, $1, $3      # registrador $2 é escrito pela instrução sub
and   $12, $2, $5     # primeiro operando ($2) depende de sub
or    $13, $6, $2     # segundo operando ($2) depende de sub
add   $14, $2, $2     # primeiro e segundo operandos ($2) dependem de sub
sw    $15, 100($2)    # base ($2) depende de sub
```

Para estudar as conseqüências destas dependências quando da execução em pipeline, usar um diagrama de pipeline de múltiplos ciclos.

2. Organizações do MIPS: pipeline

Conflitos por Dados



2. Organizações do MIPS: pipeline

Conflitos por Dados

Uma solução seria o compilador evitar seqüências de instruções que gerassem conflitos por dados

```
sub   $2, $1, $3      # registrador $2 é escrito pela instrução sub
nop                   # na falta de instruções que sejam independentes, o
nop                   # compilador inseriria instruções "nop"
and   $12, $2, $5     # primeiro operando ($2) depende de sub
or    $13, $6, $2     # segundo operando ($2) depende de sub
add   $14, $2, $2     # primeiro e segundo operandos ($2) dependem de sub
sw    $15, 100($2)    # base ($2) depende de sub
```

Problemas:

- Conflitos por dados são muito freqüentes
- A inserção de instruções nop causa perda de desempenho!

2. Organizações do MIPS: pipeline

Conflitos por Dados

Outra solução

- Detectar o conflito
- Adiantar o resultado da ULA (ou da memória de dados)

Testes para detecção de conflito (uso dos registradores de pipeline):

1a) EX/MEM.RegistradorRd = DI/EX.RegistradorRs

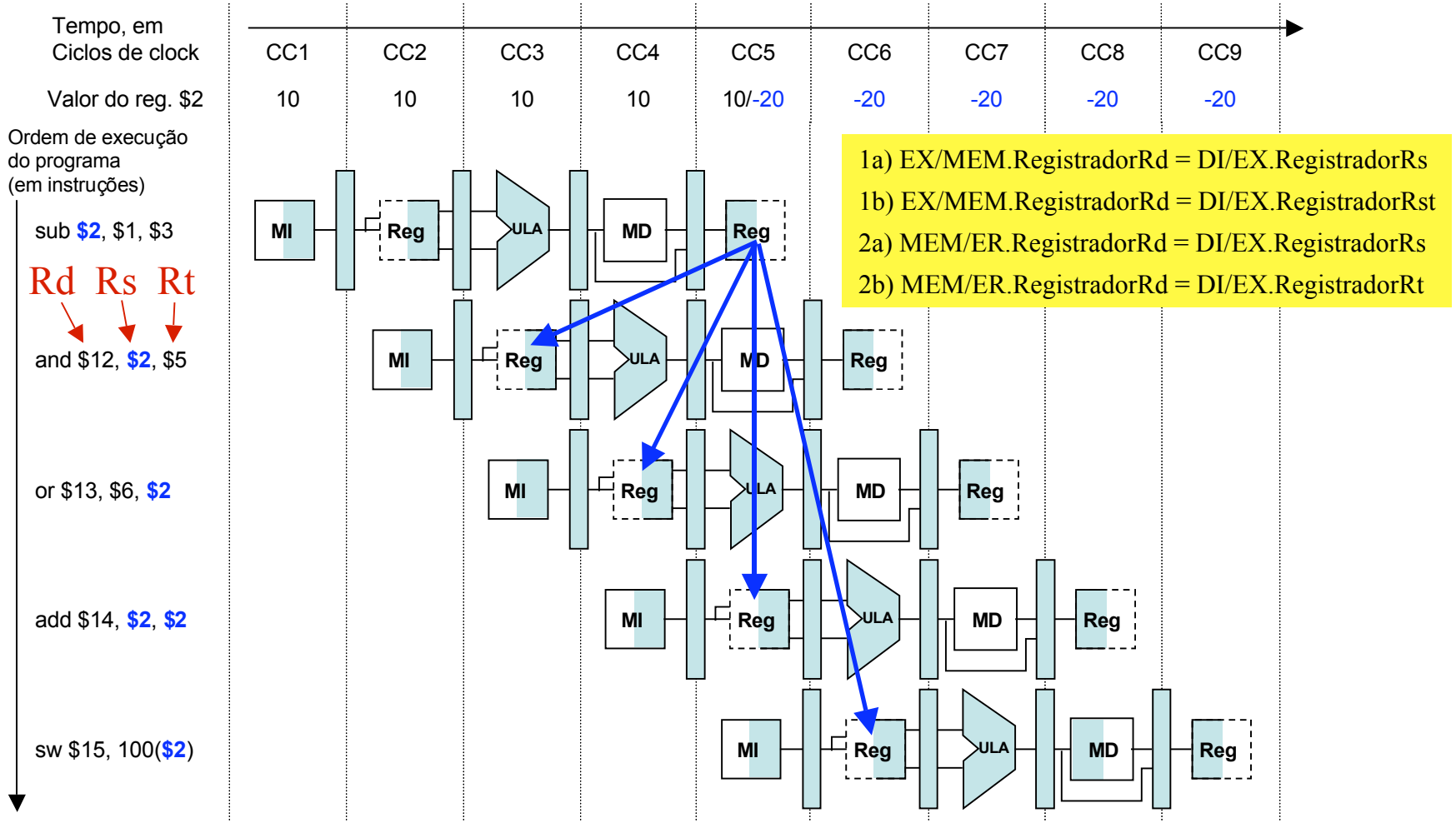
1b) EX/MEM.RegistradorRd = DI/EX.RegistradorRt

2a) MEM/ER.RegistradorRd = DI/EX.RegistradorRs

2b) MEM/ER.RegistradorRd = DI/EX.RegistradorRt

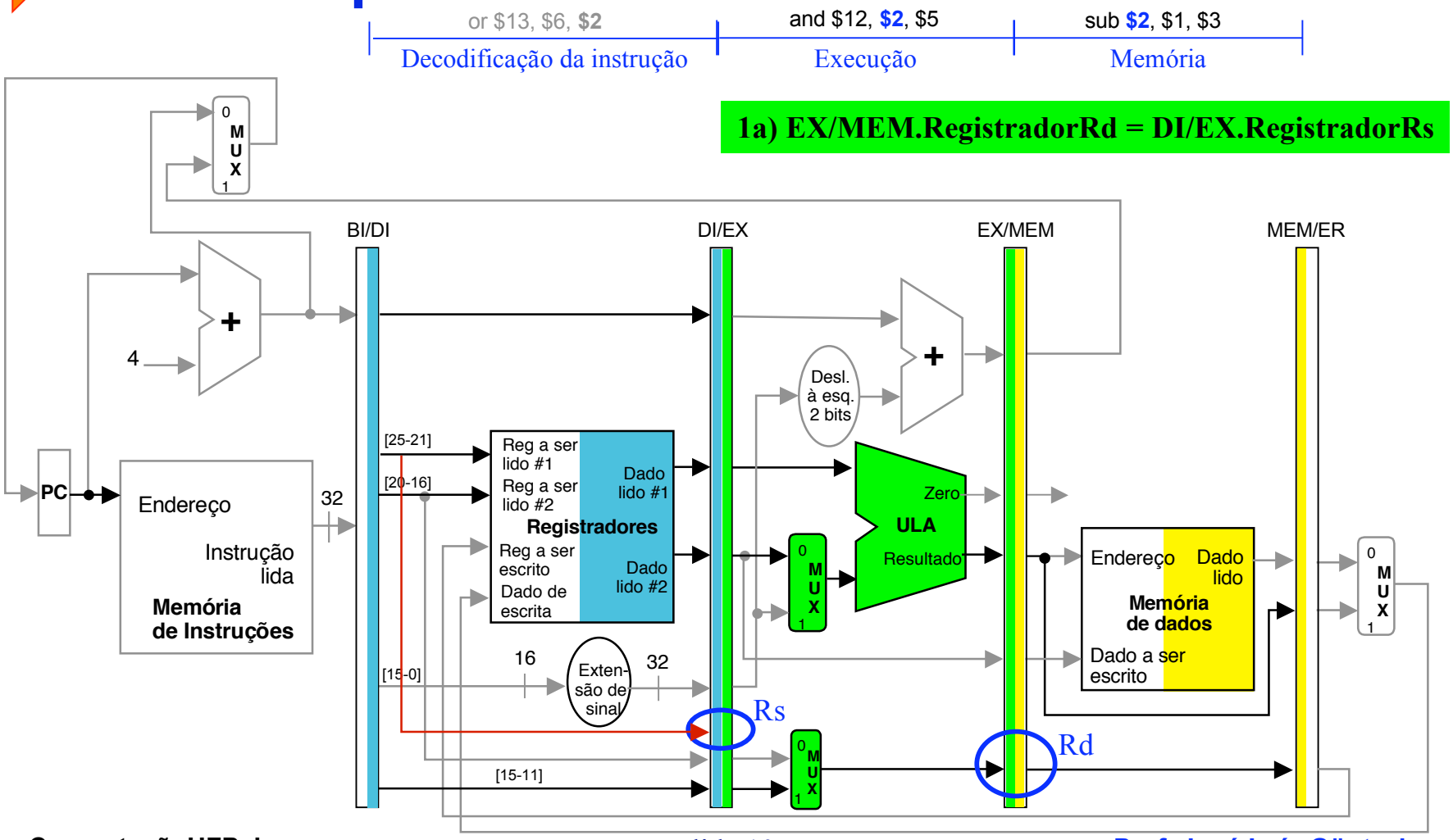
2. Organizações do MIPS: pipeline

Conflitos por Dados



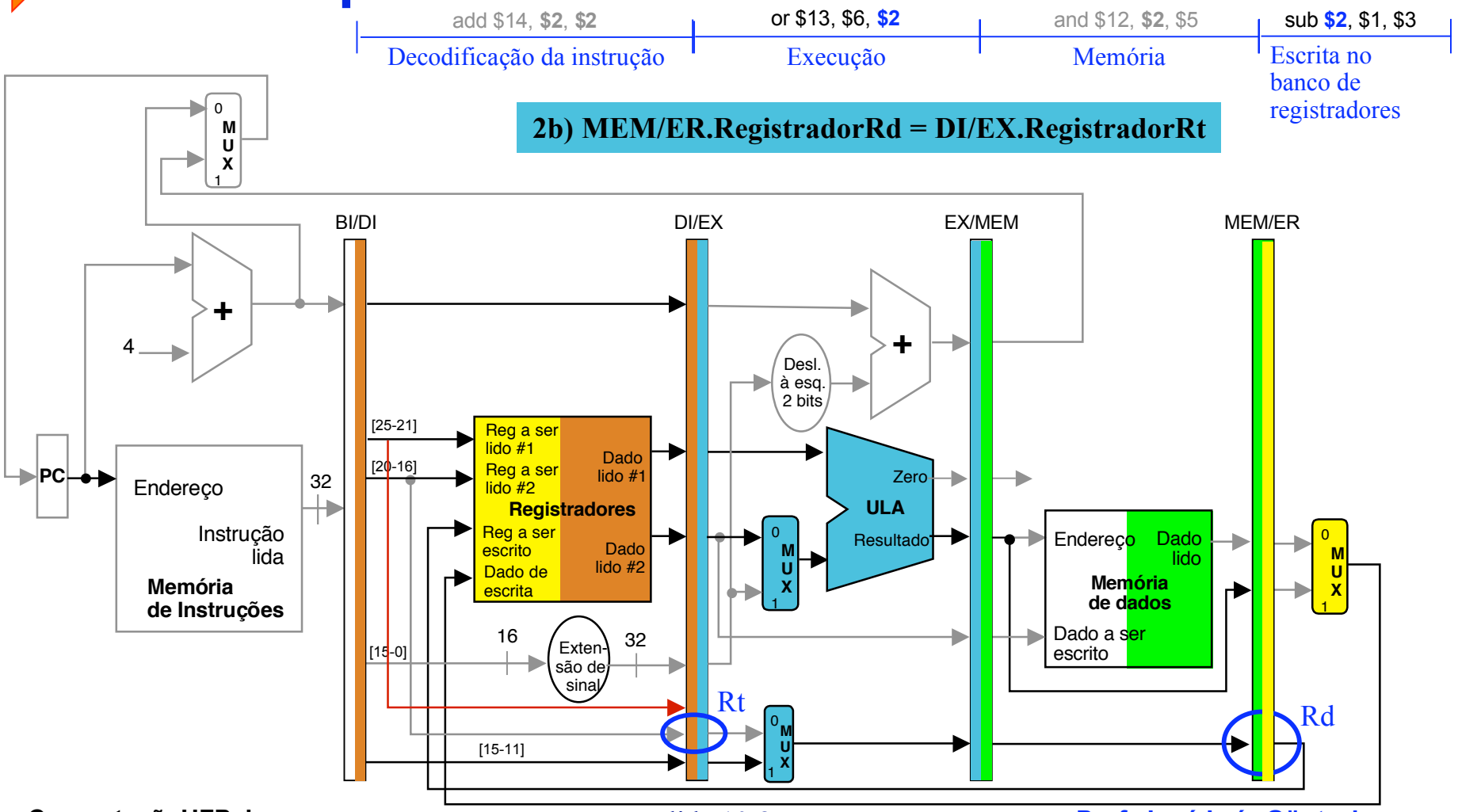
2. Organizações do MIPS: pipeline

Conflitos por Dados



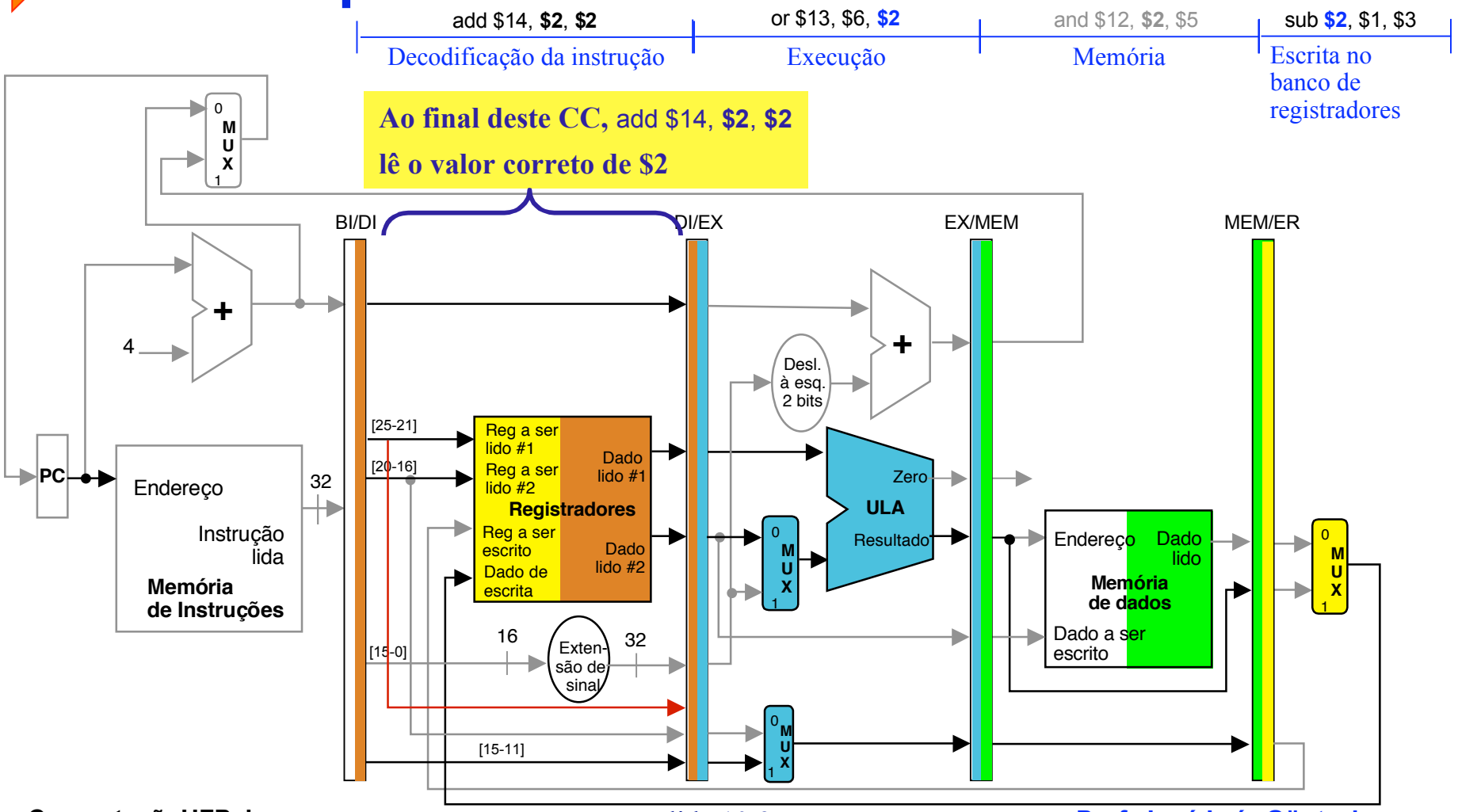
2. Organizações do MIPS: pipeline

Conflitos por Dados



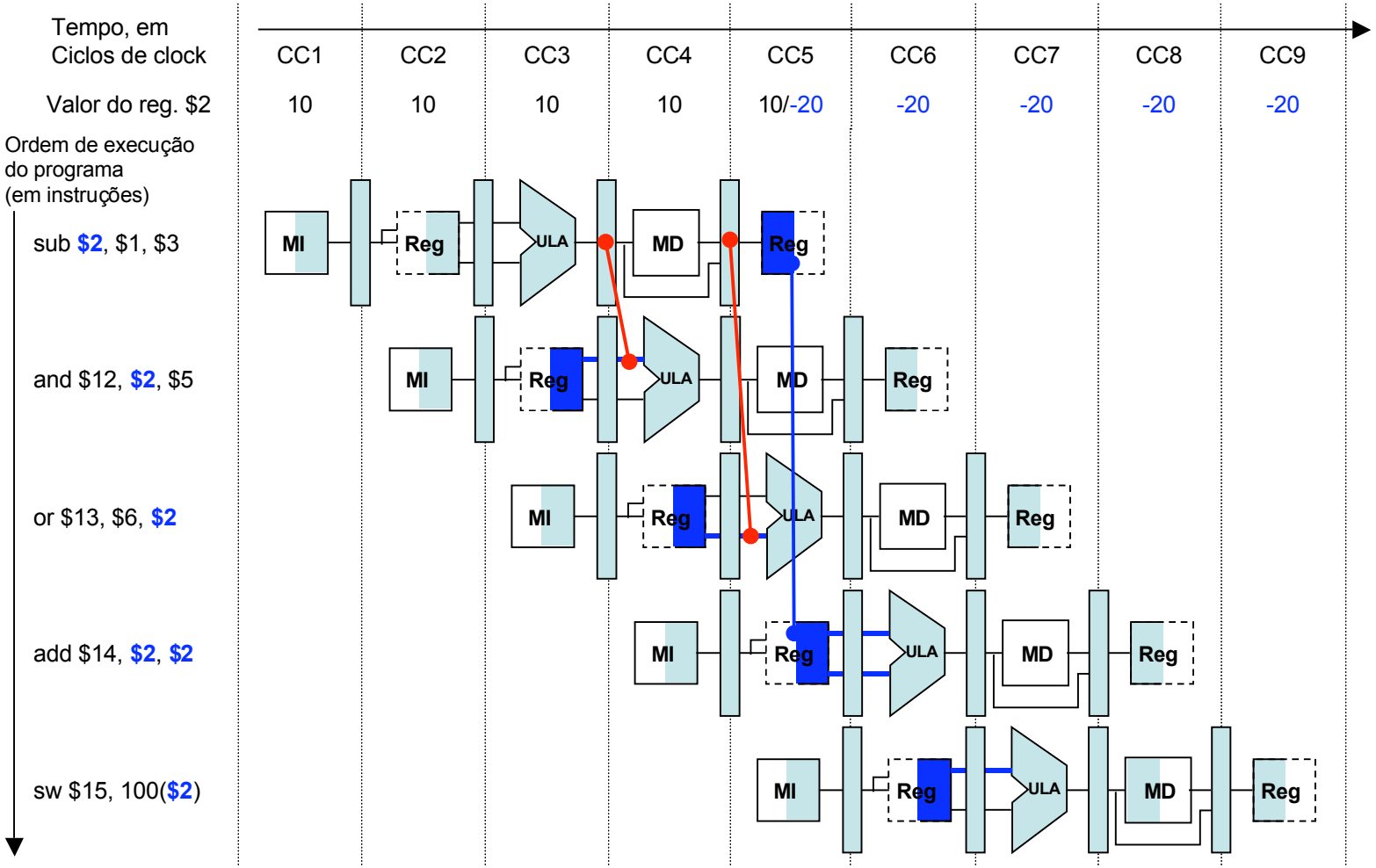
2. Organizações do MIPS: pipeline

Conflitos por Dados



2. Organizações do MIPS: pipeline

Conflitos por Dados



2. Organizações do MIPS: pipeline

Detectando e Resolvendo Conflitos por Dados

1. Conflitos no Estágio EX

se (EX/MEM.EscReg = 1
e (EX/MEM.RegistradorRd ≠ 0)
e (EX/MEM.RegistradorRd = DI/EX.RegistradorRs)) Adianta.A = 10

se (EX/MEM.EscReg = 1
e (EX/MEM.RegistradorRd ≠ 0)
e (EX/MEM.RegistradorRd = DI/EX.RegistradorRt)) Adianta.B = 10

2. Organizações do MIPS: pipeline

Detectando e Resolvendo Conflitos por Dados

2. Conflitos no Estágio MEM

se (EX/MEM.EscReg = 1
e (EX/MEM.RegistradorRd ≠ 0)
e (MEM/ER.RegistradorRd = DI/EX.RegistradorRs)) Adianta.A = 01

se (EX/MEM.EscReg = 1
e (EX/MEM.RegistradorRd ≠ 0)
e (MEM/ER.RegistradorRd = DI/EX.RegistradorRt)) Adianta.B = 01

2. Organizações do MIPS: pipeline

Detectando e Resolvendo Conflitos por Dados

Complicação:

- **Conflito entre o resultado no estágio ER e o resultado no estágio MEM e o operando-fonte da instrução no estágio da ULA.**

```
add   $1, $1, $2
add   $1, $1, $3
add   $1, $1, $4
      ...
```

2. Organizações do MIPS: pipeline

Detectando e Resolvendo Conflitos por Dados

2. Conflitos no Estágio MEM

se (EX/MEM.EscReg = 1
e (EX/MEM.RegistradorRd ≠ 0)
e (EX/MEM.RegistradorRd ≠ DI/EX.RegistradorRs)
e (MEM/ER.RegistradorRd = DI/EX.RegistradorRs)) Adianta.A = 01

se (EX/MEM.EscReg = 1
e (EX/MEM.RegistradorRd ≠ 0)
e (EX/MEM.RegistradorRd ≠ DI/EX.RegistradorRt)
e (MEM/ER.RegistradorRd = DI/EX.RegistradorRt)) Adianta.B = 01

2. Organizações do MIPS: pipeline

Uso do Simulador DLX

- O processador DLX é um primo-irmão do MIPS, mostrado no outro livro do Patersson-Hennessy (**Computer Architecture: A Quantitative Approach. 2nd edition. San Francisco, California: Morgan Kaufmann Publishers, 1996.**)
- O DLX é capaz de executar todas as instruções do MIPS
- Disponível em **ftp://ftp.mkp.com/pub/dlx/**
- O arquivo-fonte contendo o programa em simbólico deve ser editado em um editor ascii, sem formatação, e deve receber um nome com extensão ".s"
- O simulador do DLX mostra a progressão das instruções nos estágios do pipeline
- Permite visualizar a execução com e sem Unidade de Adiantamento (*with or without forwarding*)