Universidade Federal de Pelotas
Centro de Desenvolvimento Tecnológico
Bacharelado em Ciência da Computação
Engenharia de Computação

Arquitetura e Organização de Computadores I

Aula 11

2. MIPS pipeline: conflitos por dados e adiantamento, paradas e conflito de controle.

Prof. Guilherme Corrêa
gcorrea@inf.ufpel.edu.br

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 consequências destas dependências quando da execução em pipeline, usar um diagrama de pipeline de múltiplos ciclos.

2. Organizações do MIPS: pipeline

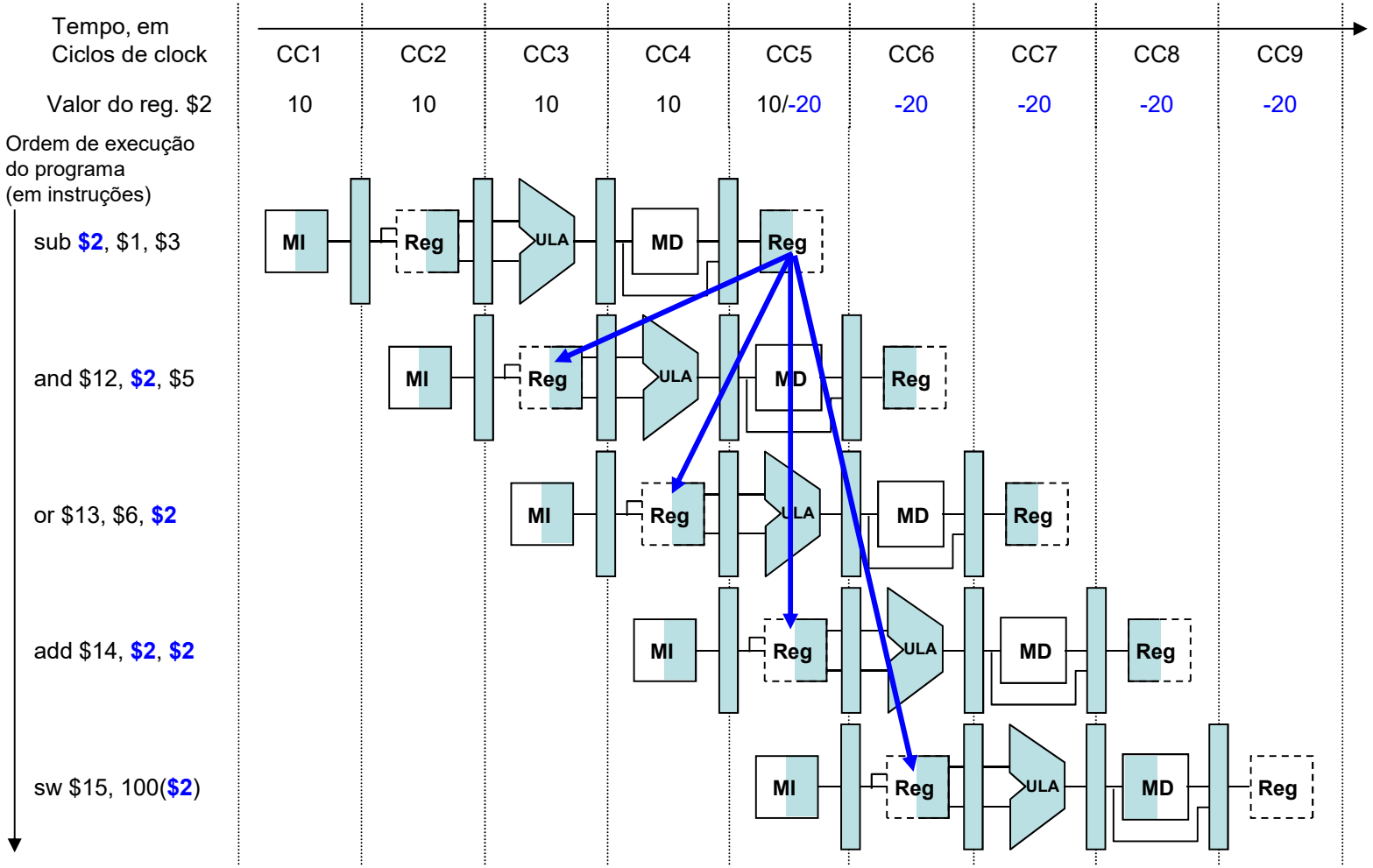
Conflitos por Dados

Considerando que:

- \$2 possui o valor **10** antes da execução da instrução sub
- \$2 assume o valor **-20** após a execução da instrução sub
- O projeto do banco de registradores permite **uma leitura e uma escrita no mesmo ciclo**
  - Escrita na primeira metade do ciclo e leitura na metade final

2. Organizações do MIPS: pipeline

Conflitos por Dados



2. Organizações do MIPS: pipeline

Conflitos por Dados

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

- **Compilador identifica os conflitos e os evita!**
- **Como pode ser feito?**

```
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
```

2. Organizações do MIPS: pipeline

Conflitos por Dados

Uma solução seria o compilador evitar sequências 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 frequentes
- 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)

Notação para os registradores:

EX/MEM.RegistradorRd = DI/EX.RegistradorRs

2. Organizações do MIPS: pipeline

Conflitos por Dados

Outra solução

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

Tipos de conflitos:

Tipo 1 – Estágio de execução em relação ao estágio de leitura dos registradores

Tipo 2 – Estágio de acesso a memória em relação ao estágio de leitura de registradores

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

Outra solução

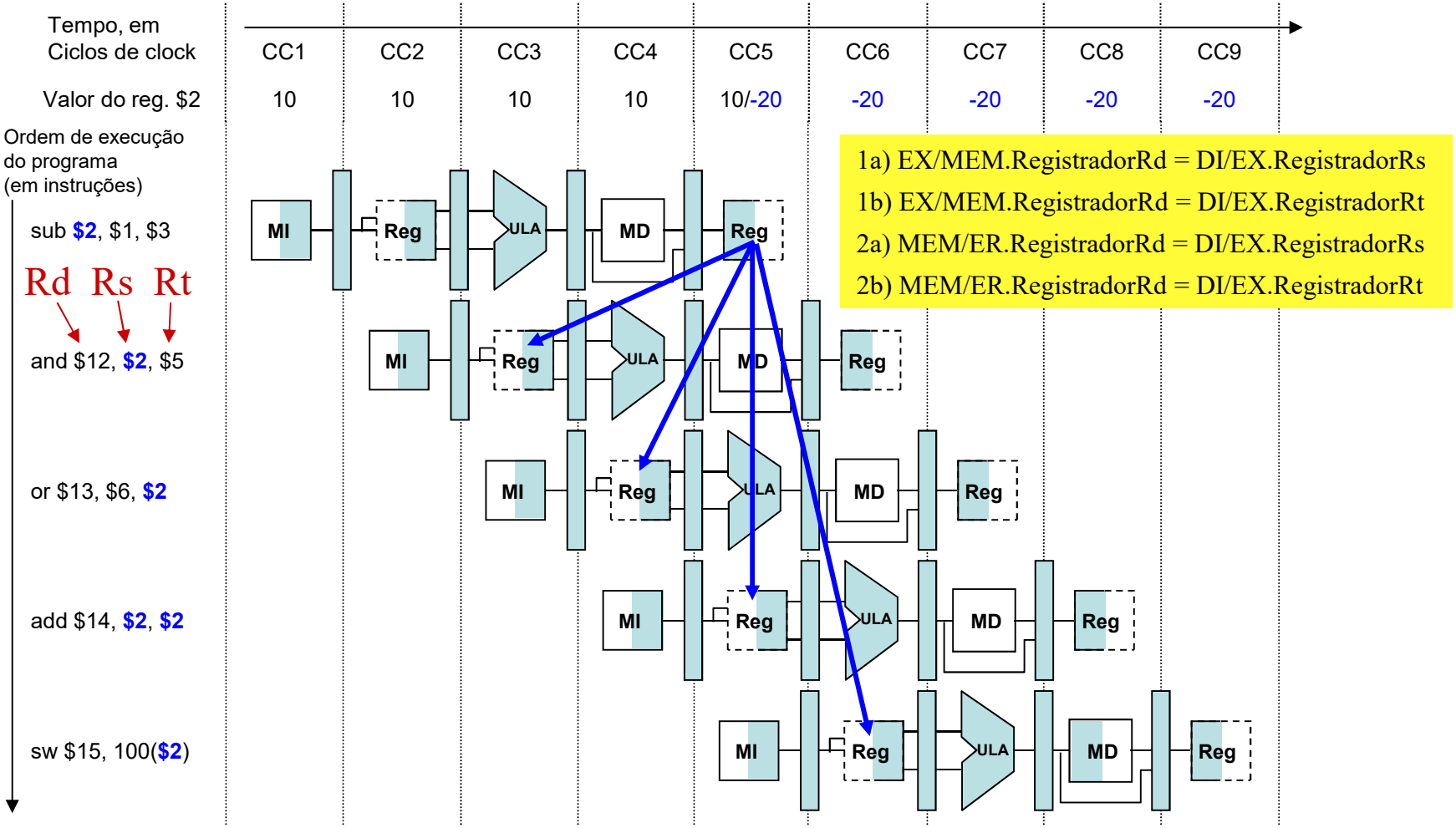
- **Detectar o conflito**
- **Adiantar o resultado da ULA (ou da memória de dados)**

Exemplo: Classifique as dependências de dados existentes no código como 1a, 1b, 2a ou 2b:

```
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
```

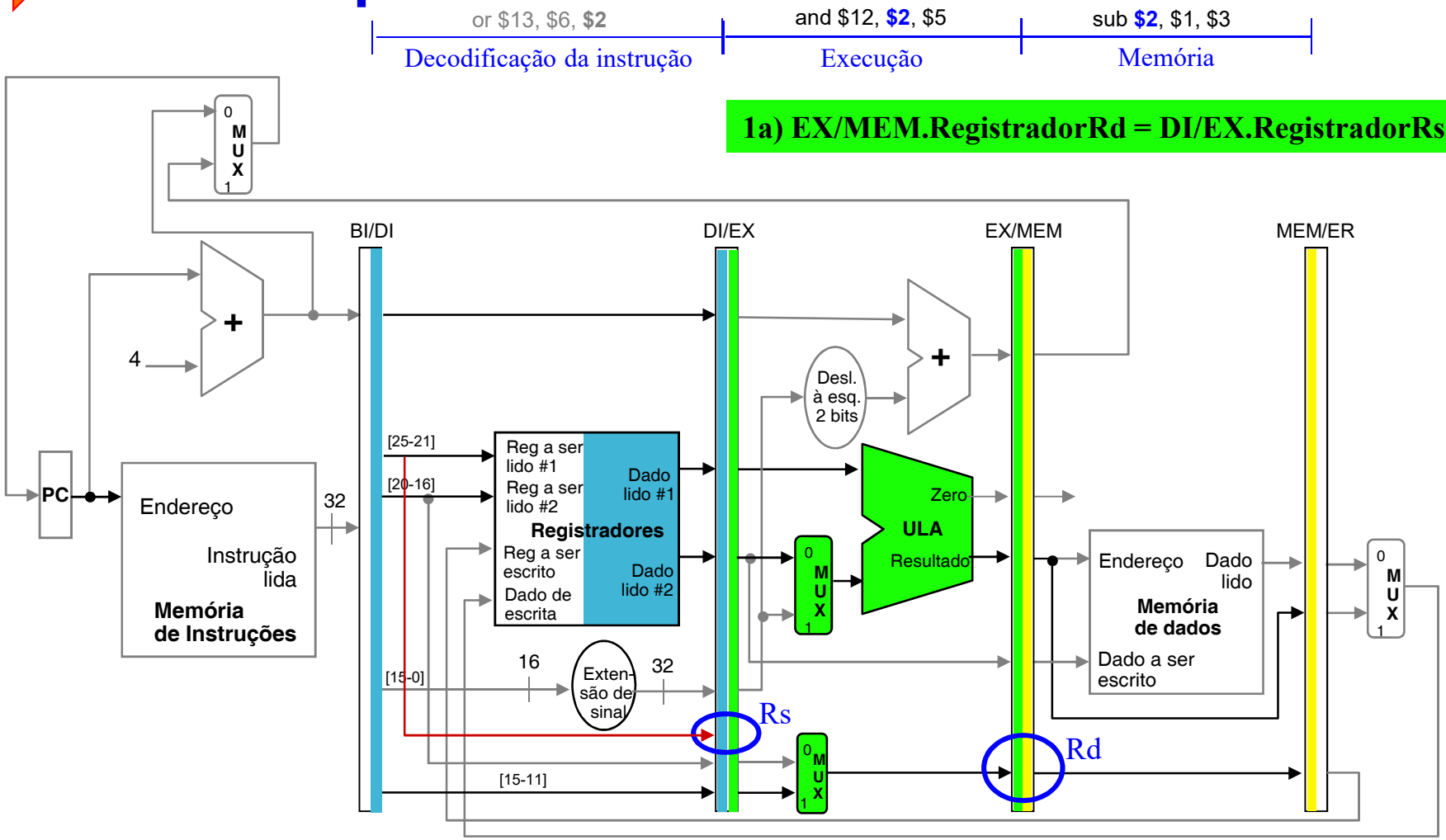
2. Organizações do MIPS: pipeline

Conflitos por Dados



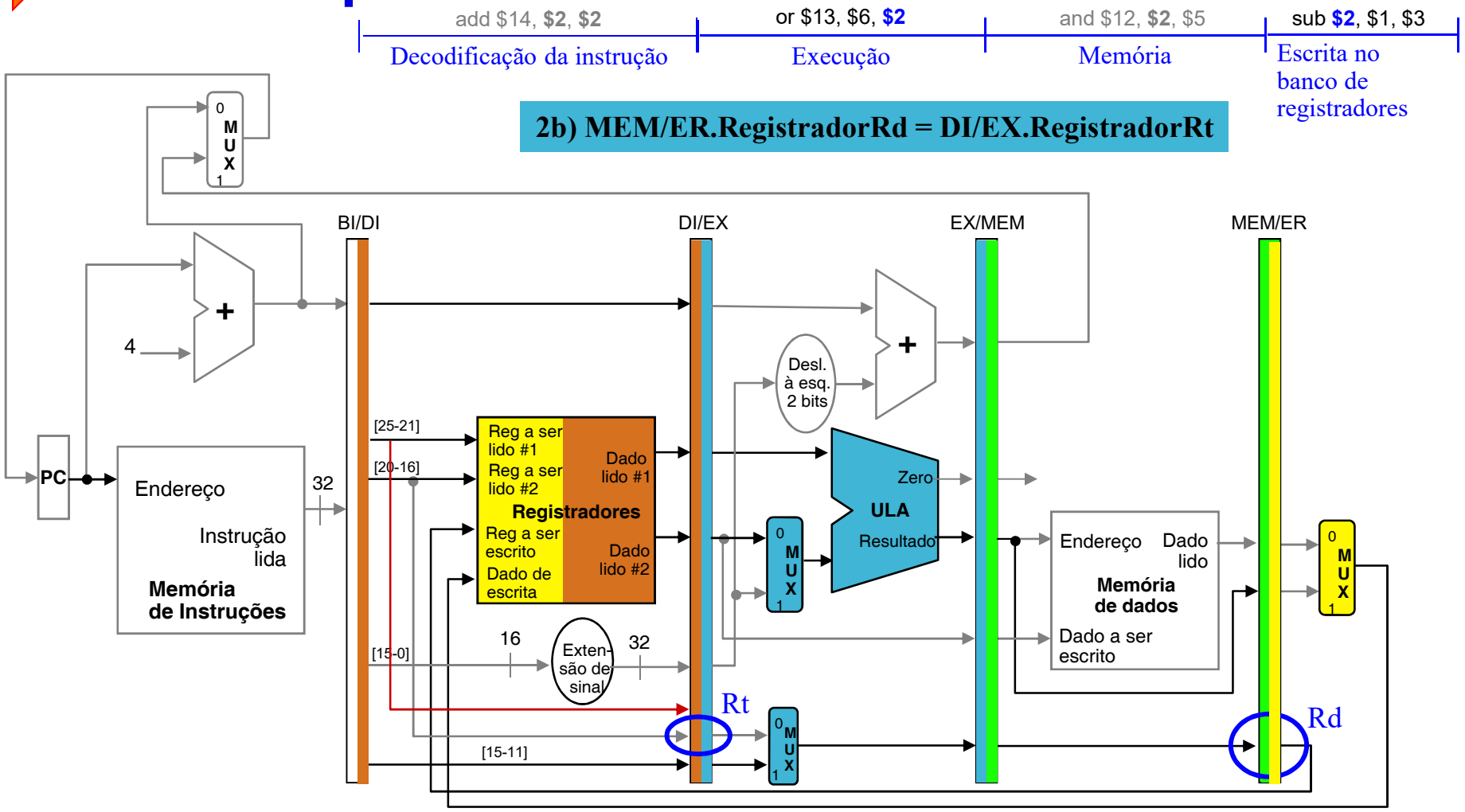
2. Organizações do MIPS: pipeline

Conflitos por Dados



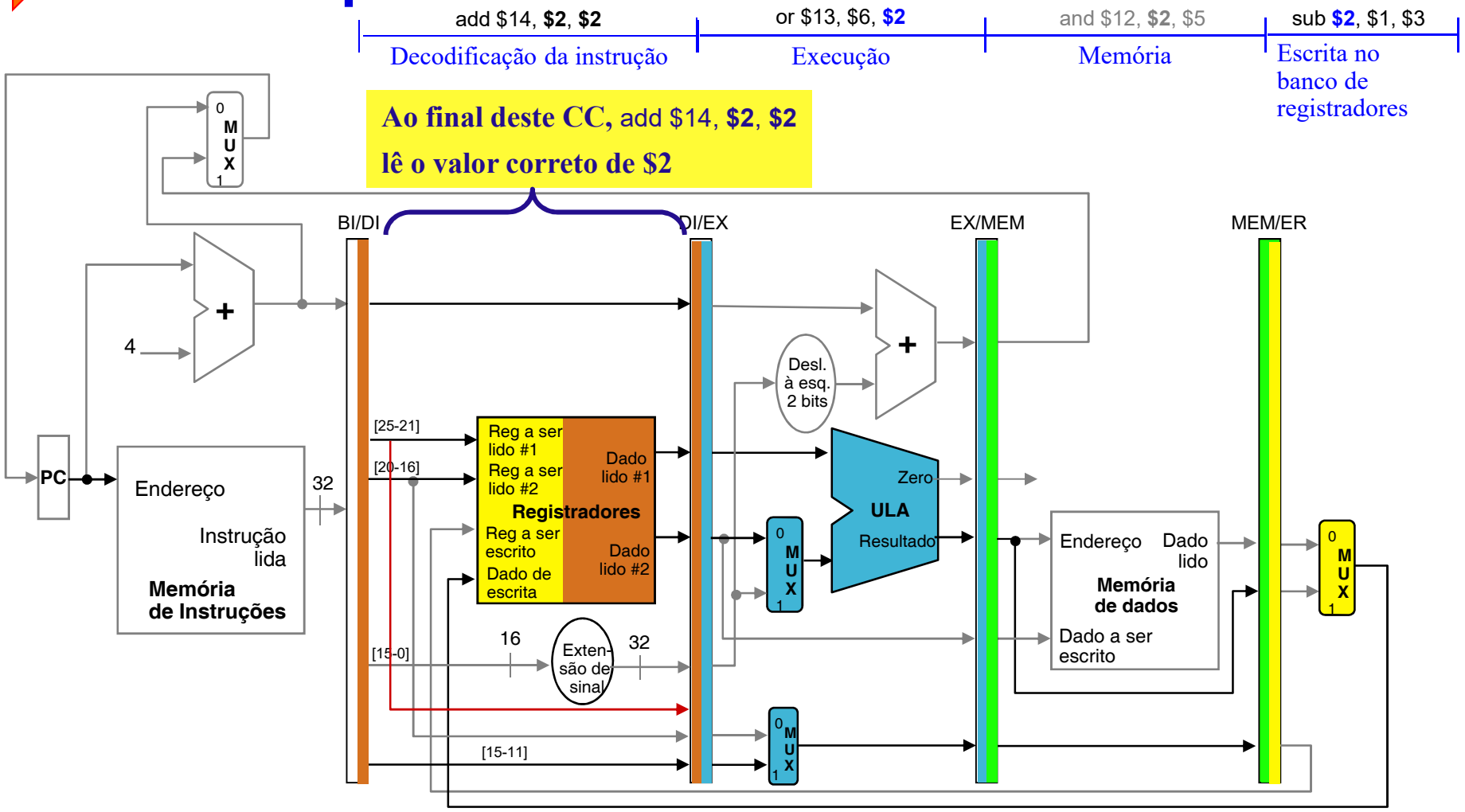
2. Organizações do MIPS: pipeline

Conflitos por Dados



2. Organizações do MIPS: pipeline

Conflitos por Dados



2. Organizações do MIPS: pipeline

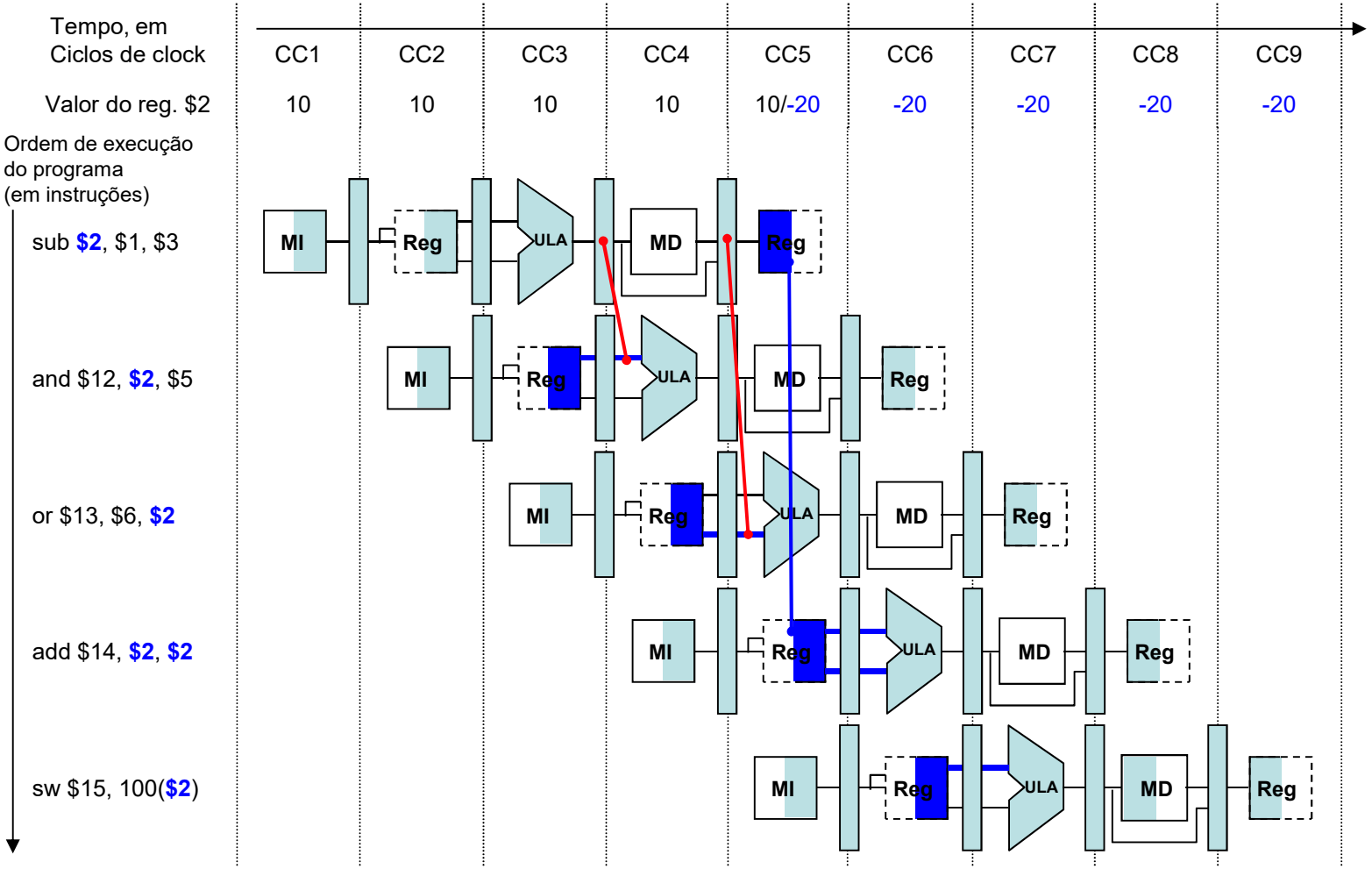
Conflitos por Dados

Considerando apenas instruções do tipo R:

- Os dados que precisamos nas entradas da ULA já estão disponíveis nos registradores do pipeline
- Se pudermos obter as entradas da ULA de qualquer dos registrados do pipeline, e não apenas do DI/EX, podemos resolver todos os conflitos de dados!
- Qual o custo adicional para isso?

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

se (MEM/ER.EscReg = 1
e (MEM/ER.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
      …
```

- **Neste caso, o Rs ($1) da instrução 3 não tem dependência real com o Rd ($1) da instrução 1 (dep. tipo 2). A dependência real é com o Rd ($1) da instrução 2 (dep. tipo 1).**
- **Resolver primeiro os conflitos do tipo 1.**

2. Organizações do MIPS: pipeline

Detectando e Resolvendo Conflitos por Dados

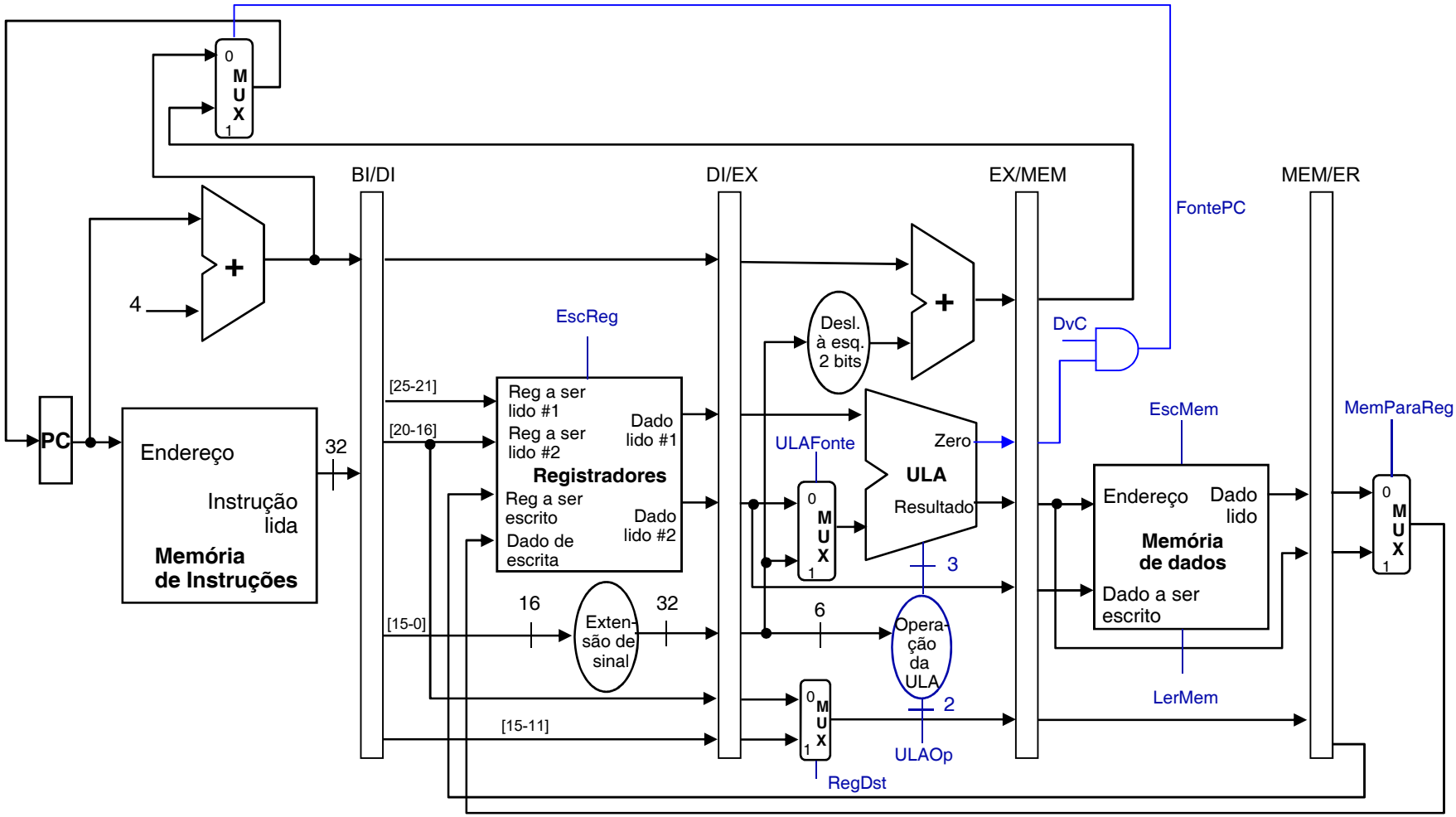
2. Conflitos no Estágio MEM

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

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

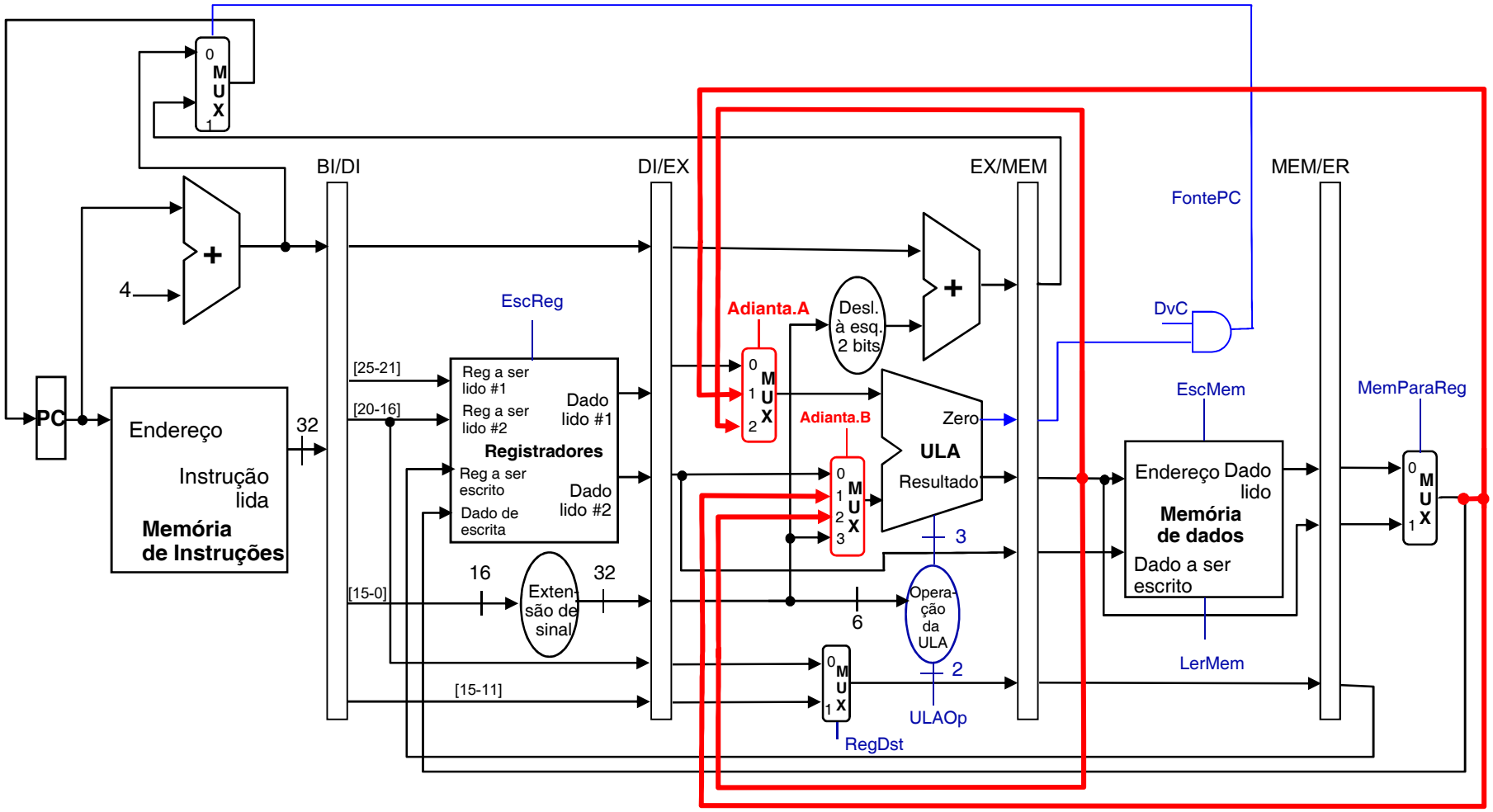
2. Organizações do MIPS: pipeline

Bloco Operativo em Pipeline com Sinais de Controle



2. Organizações do MIPS: pipeline

Bloco Operativo em Pipeline com Adiantamento 1 e 2



2. Organizações do MIPS: pipeline

Detectando e Resolvendo Conflitos por Dados
Unidade de adiantamento:

Controle do Multiplexador	Fonte	Explicação
Adianta.A = 00	DI/EX	O primeiro operando da ULA vem do banco de registradores
Adianta.A = 10	EX/MEM	O primeiro operando da ULA é adiantado a partir do seu resultado anterior
Adianta.A = 01	MEM/ER	O primeiro operando da ULA é adiantado a partir da memória de dados ou de um resultado anterior da ULA
Adianta.B = 00	DI/EX	O segundo operando da ULA vem do banco de registradores
Adianta.A = 10	EX/MEM	O segundo operando da ULA é adiantado a partir do seu resultado anterior
Adianta.A = 01	MEM/ER	O segundo operando da ULA é adiantado a partir da memória de dados ou de um resultado anterior da ULA

2. Organizações do MIPS: pipeline

Detectando e Resolvendo Conflitos por Dados

Exercício:

Mostre como a unidade de adiantamento de dados pode resolver as dependências presentes no código abaixo:

sub **$2**, $1, $3
and **$4, $2**, $5
or **$4, $4, $2**
add $9, **$4, $2**

Indique o tipo de dependência e os ciclos de clock onde as dependências ocorrem, bem como a solução adequada para resolver o conflito.

2. Organizações do MIPS: pipeline

Detectando e Resolvendo Conflitos por Dados

Exercício: Resolução

Conflito entre as instruções 2 e 1: Tipo 1, ciclo 4
- Solução: Adianta a entrada superior da ULA do resultado anterior da ULA (EX/MEM ou Adianta.A = 10)

Conflito entre as instruções 3 e 2: Tipo 1, ciclo 5
- Solução: Adianta a entrada superior da ULA do resultado anterior da ULA (EX/MEM ou Adianta.A = 10)

Conflito entre as instruções 3 e 1: Tipo 2, ciclo 5
- Solução: Adianta a entrada inferior da ULA do resultado do estágio MEM/ER, ou Adianta.B = 01)

Conflito entre as instruções 4 e 3: Tipo 1, ciclo 6
- Solução: Adianta a entrada superior da ULA do resultado do estágio EX/MEM, ou Adianta.B = 10)
- Solução: A entrada inferior da ULA não é adiantada

2. Organizações do MIPS: pipeline

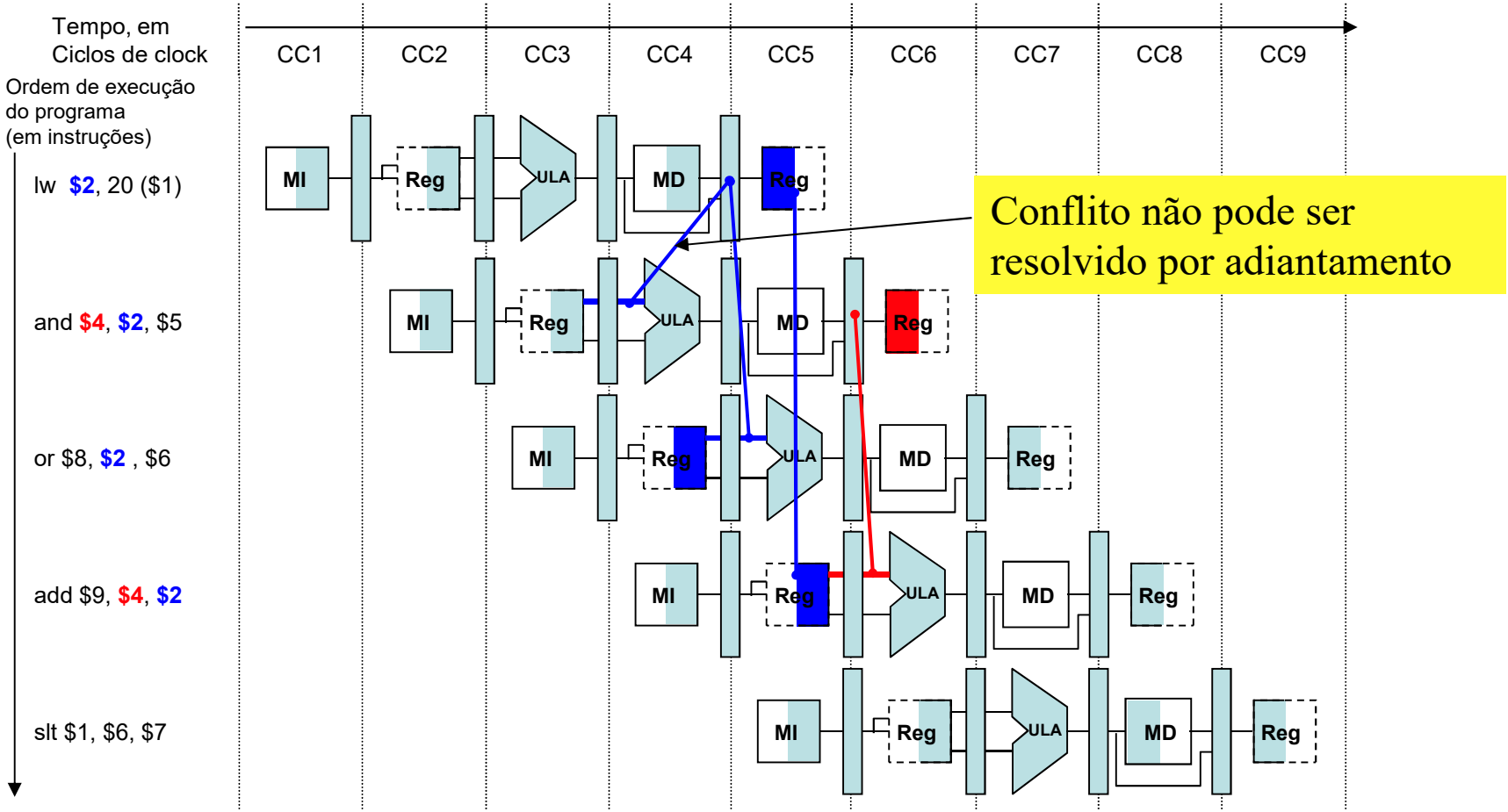
Conflitos por Dados e Paradas

Nem sempre o adiantamento irá resolver um conflito por dados. Exemplo:

```
lw    $2, 20($1)      # registrador $2 é escrito
and   $4, $2, $5      # primeiro operando ($2) depende de lw; registrador $4 é escrito
or    $8, $2 , $6     # primeiro operando ($2) depende de lw
add   $9, $4, $2      # primeiro operando ($4) depende de and; segundo operando ($2)
                      depende de lw
slt   $1, $6, $7      # nenhuma dependencia
```

2. Organizações do MIPS: pipeline

Conflitos por Dados e Paradas



2. Organizações do MIPS: pipeline

Conflitos por Dados e Paradas

Unidade de Detecção de Conflito

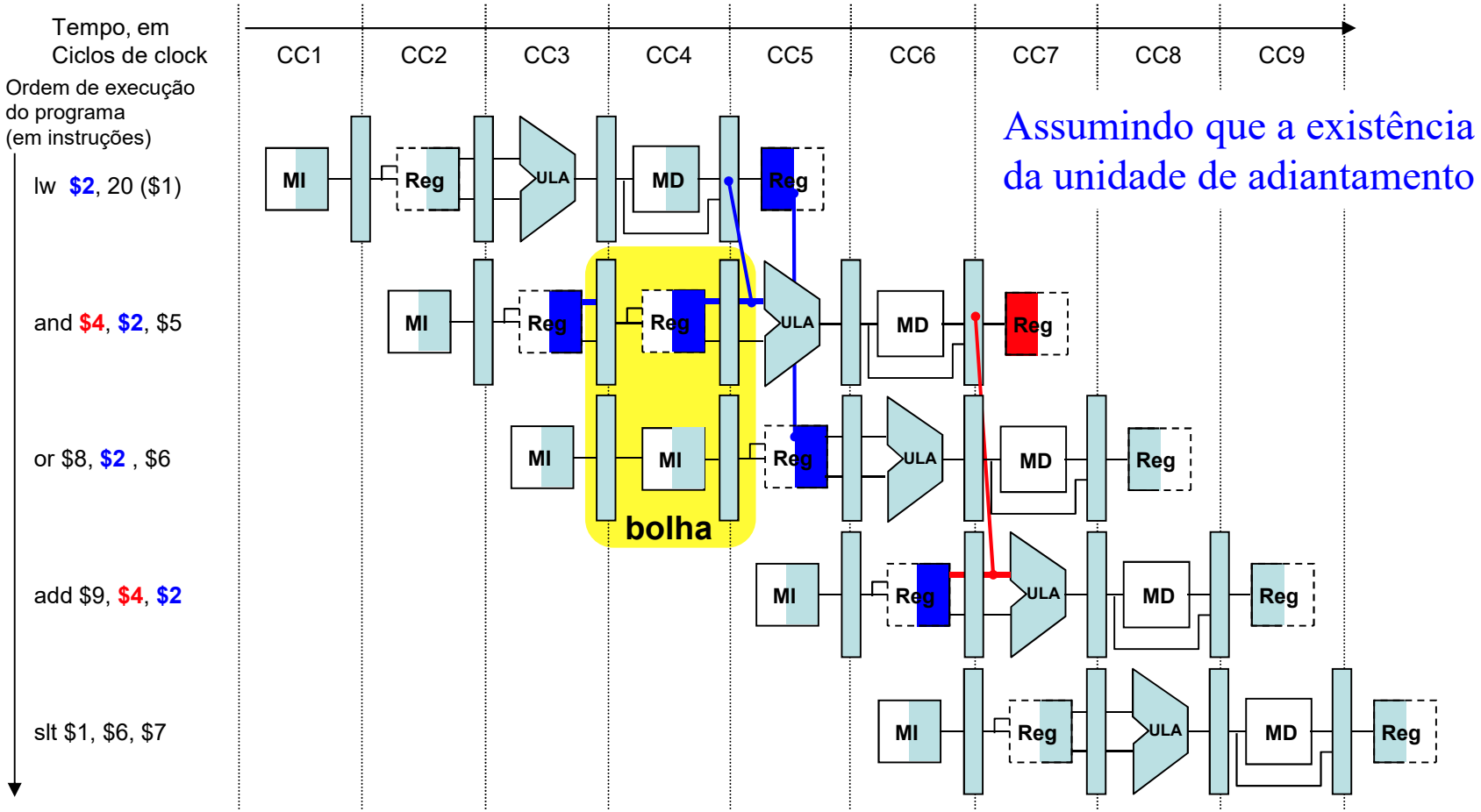
- **Faz o pipeline parar quando houver uma instrução load word, seguida de uma instrução que leia o registrador onde esta instrução de load word escreveu**
- **Vai operar durante o estágio DI, inserindo uma parada entre a instrução load word e o uso de seu resultado**
- **Condição a ser verificada:**

load word é a única instrução que lê dados da memória

Se (DI/EX.LerMem = 1 E
((DI/EX.RegistradorRd = BI/DI.RegistradorRs) OU
(DI/EX.RegistradorRd= BI/DI.RegistradorRt)))
Então pára o pipeline por um ciclo de relógio

2. Organizações do MIPS: pipeline

Conflitos por Dados e Paradas



2. Organizações do MIPS: pipeline

Conflitos por Dados e Paradas

Trancando o Prosseguimento das Instruções Posteriores a uma Instrução "load word"

- **Se a instrução que está no estágio DI estiver parada, então o estágio BI também precisa parar**
- **Para impedir o avanço de instruções pelo pipeline, basta evitar que tanto o PC quanto o registrador BI/DI sejam escritos**
- **As condições do item anterior fazem com que, no ciclo de relógio seguinte:**
  - A instrução que está no BI seja lida novamente
  - Os registradores lidos no estágio DI serão lidos novamente

2. Organizações do MIPS: pipeline

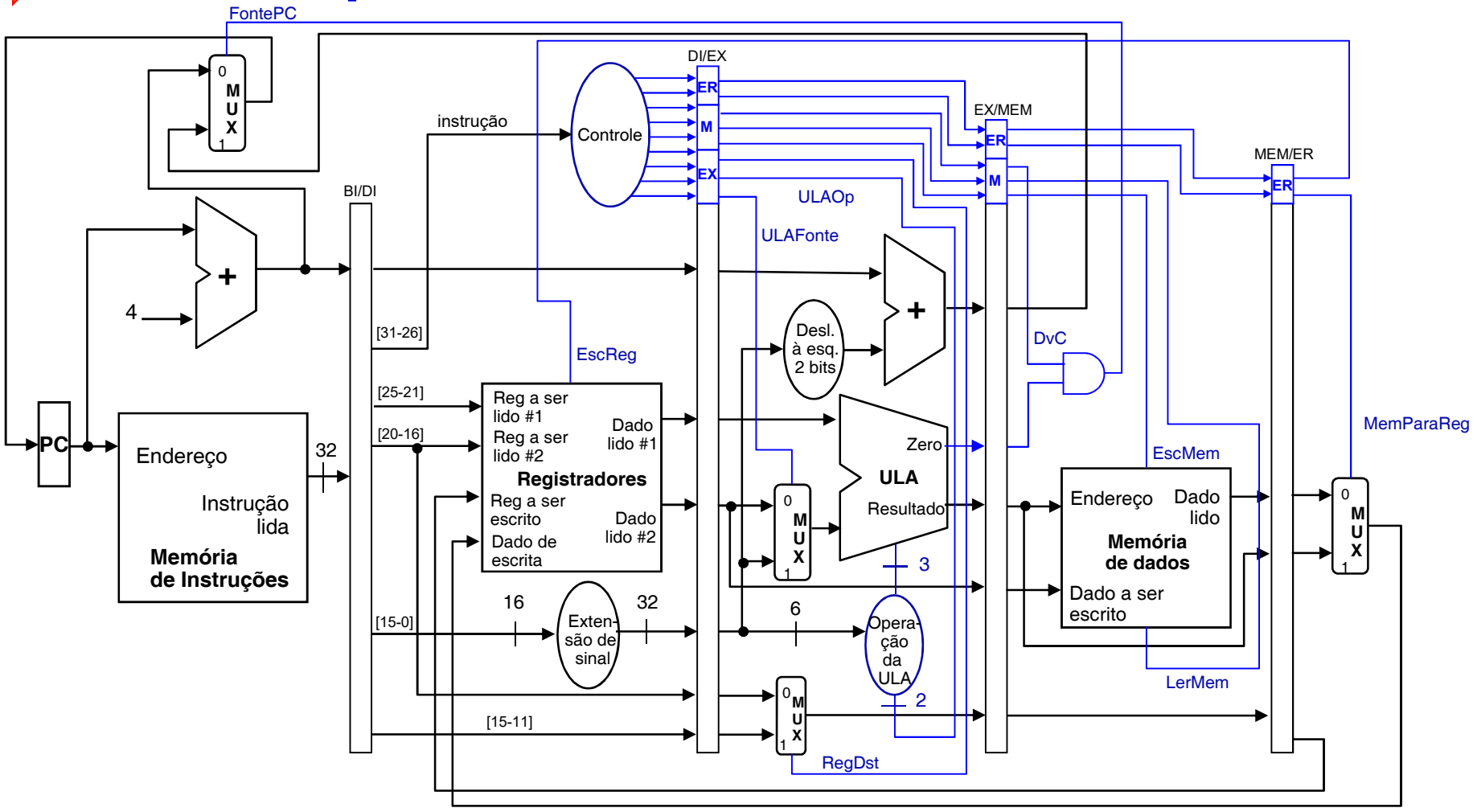
Conflitos por Dados e Paradas

Propagando uma Bolha pelo Pipeline

- **Como a instrução load word prossegue pelo pipeline, cria-se uma "bolha" de execução, a qual deve também prosseguir pelo pipeline**
- **Uma "bolha" deve executar em cada estágio a mesma coisa que uma instrução NOP executa**
- **NOP:**
  - Todos os sinais de controle em 0 (zero) para os estágios EX, MEM e ER.
  - Estes valores de sinais de controle são passados adiante a cada ciclo de relógio, produzindo o efeito desejado (nenhum registrador ou memória é escrito)

2. Organizações do MIPS: pipeline

Conflitos por Dados e Paradas



2. Organizações do MIPS: pipeline

Conflitos de Controle
(ou Conflitos de Desvios Condicionais)

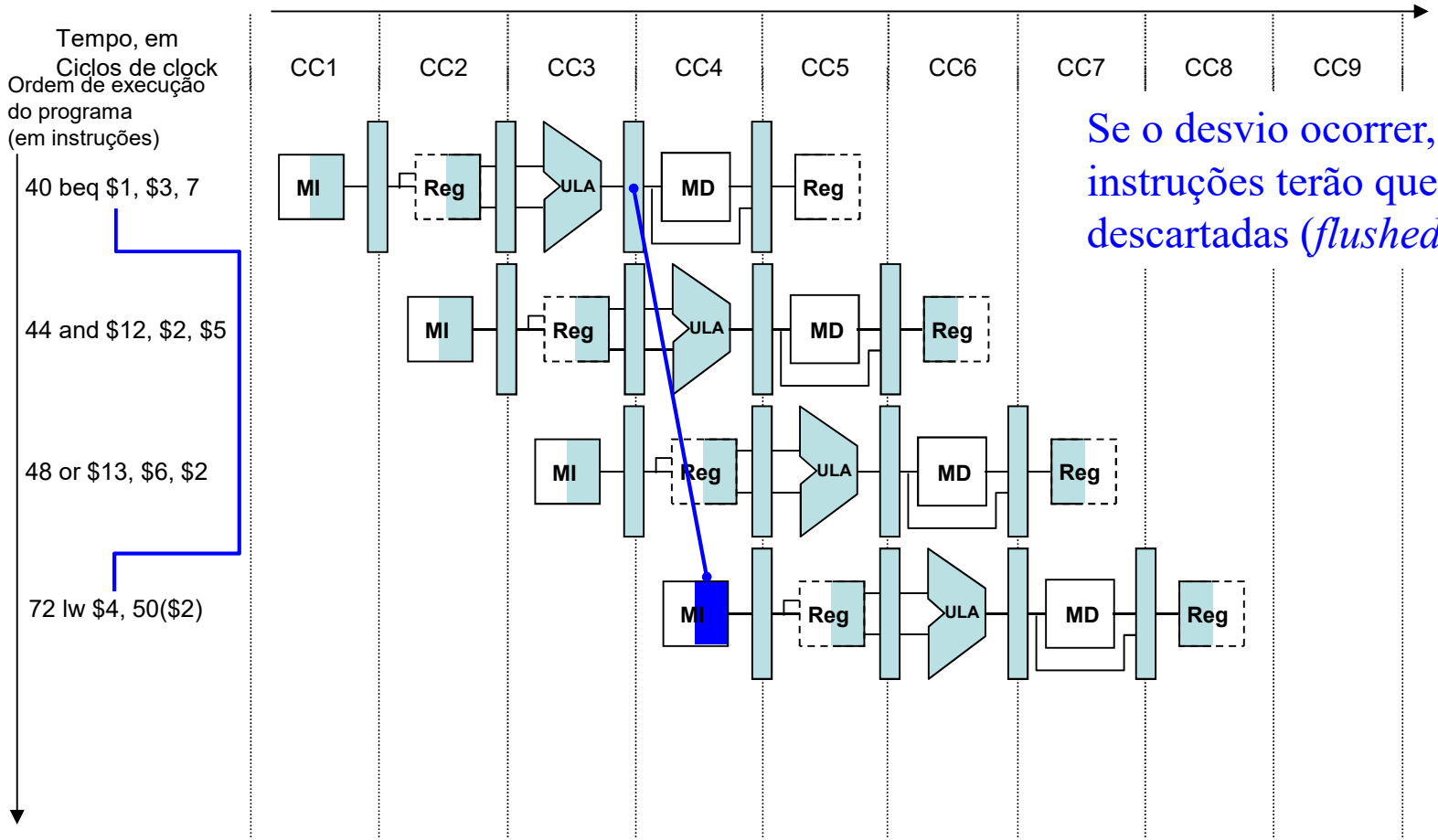
Desvio Condicional em Pipeline.
Exemplo:

```
36    sub    $10, $4, $8
40    beq    $1, $3, 7      # desvio relativo ao PC para 40 + 4 + 7*4 = 72
44    and    $12, $2, $5
48    or     $13, $2, $6
52    add    $14, $4, $2
56    slt    $15, $6, $7
…     …
72    lw     $4, 50( $7)
```

2. Organizações do MIPS: pipeline

Conflitos de Controle

Considerando o Bloco Operativo Pipeline Visto Até Aqui…



2. Organizações do MIPS: pipeline

Conflitos de Controle: Solução 1

- Conforme já visto anteriormente, a parada no avanço das instruções não é uma solução viável para o desvio condicional
- Uma alternativa comum é considerar que os desvios condicionais sempre ou nunca ocorrem, considerando a sequência normal de execução das instruções
- É mais fácil para o projeto do pipeline considerar que o desvio nunca ocorre
- Neste caso, caso o desvio se realize, será necessário descartar as instruções que estiverem sendo buscadas e executadas
- E a execução deve continuar a partir da instrução armazenada no endereço-alvo do desvio condicional…

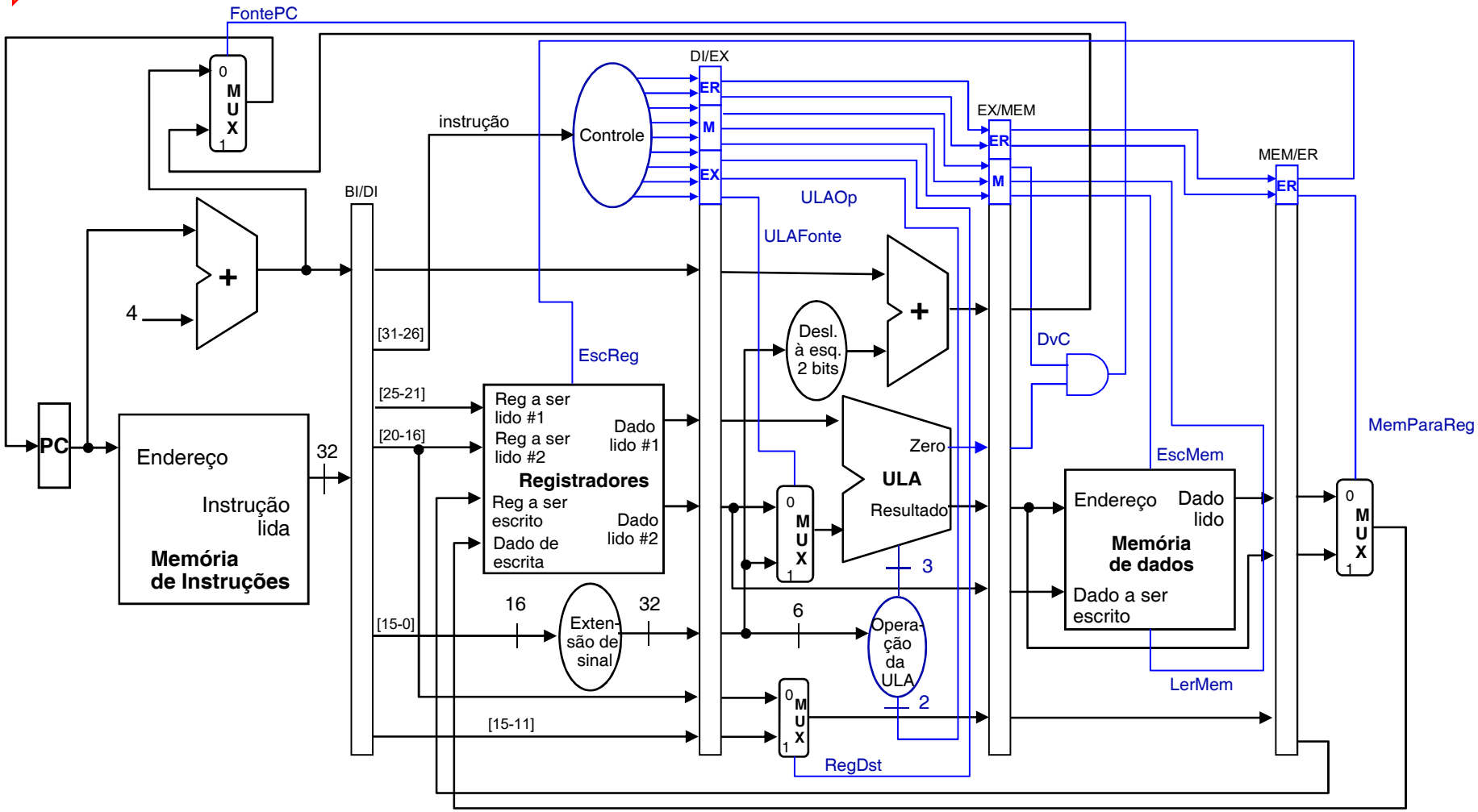
2. Organizações do MIPS: pipeline

Conflitos de Controle: Solução 1

- Para descartar instruções, basta mudar para 0 os valores originais dos sinais de controle e
- Também mudar as instruções que estiverem em BI, DI e EX, quando a instrução de desvio condicional chegar ao estágio MEM

2. Organizações do MIPS: pipeline

Conflitos de Controle



2. Organizações do MIPS: pipeline

Conflitos de Controle: Solução 2

- **Reduzir o retardo dos desvios:**
  - Selecionar o endereço de desvio ao final do estágio de EX
  - Utilizar hardware adicional para calcular o endereço de salto no estágio DI
  - Controle adicional para zerar o registrador BI

2. Organizações do MIPS: pipeline

Conflitos de Controle: Solução 2

- **Predição dinâmica de desvios:**
  - Usar um hardware adicional para manter um histórico das ocorrências, buffer de predição de desvios
  - Um bit para indicar se o desvio anterior ocorreu
  - Não afeta o tratamento do erro

2. Organizações do MIPS: pipeline

Bibliografia recomendada

- PATTERSON, David A.; HENESSY, John L. **Organização e Projeto de Computadores: a interface hardware/software**. 2ª.ed. Rio de Janeiro: LTC, 2000.