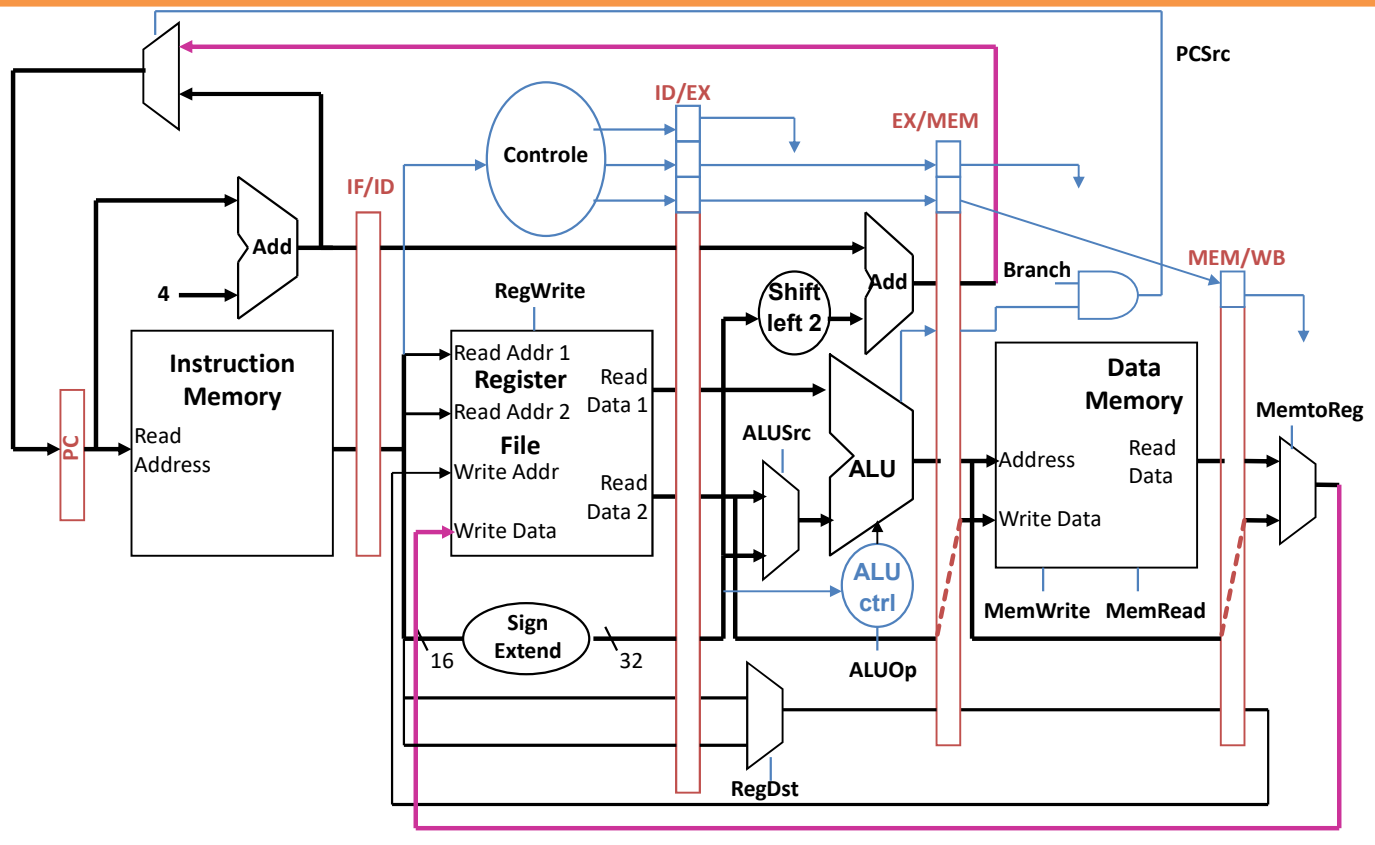
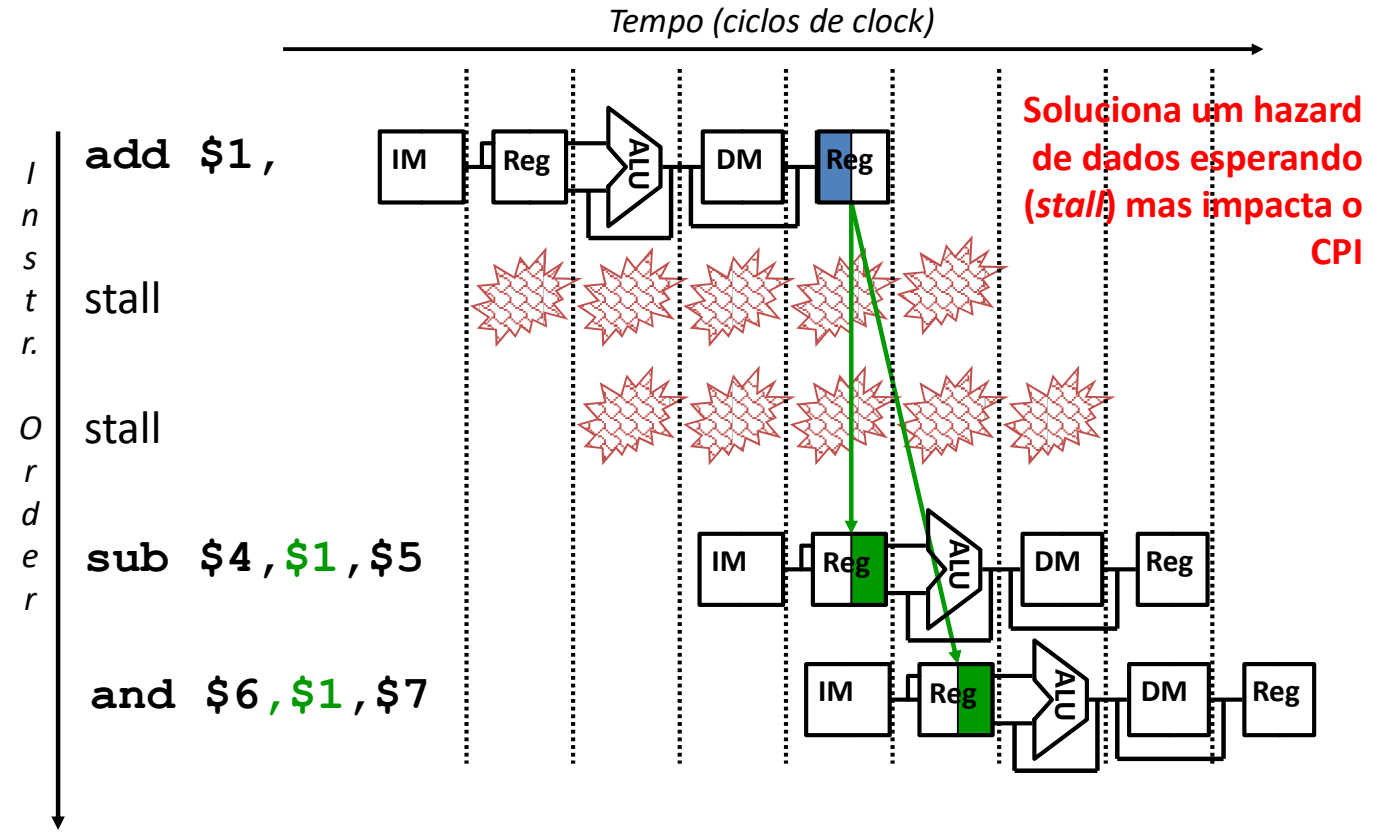
[Aula 12] Pipeline do MIPS 3: *Solucionando hazards de dados*

Prof. João F. Mari
joaof.mari@ufv.br

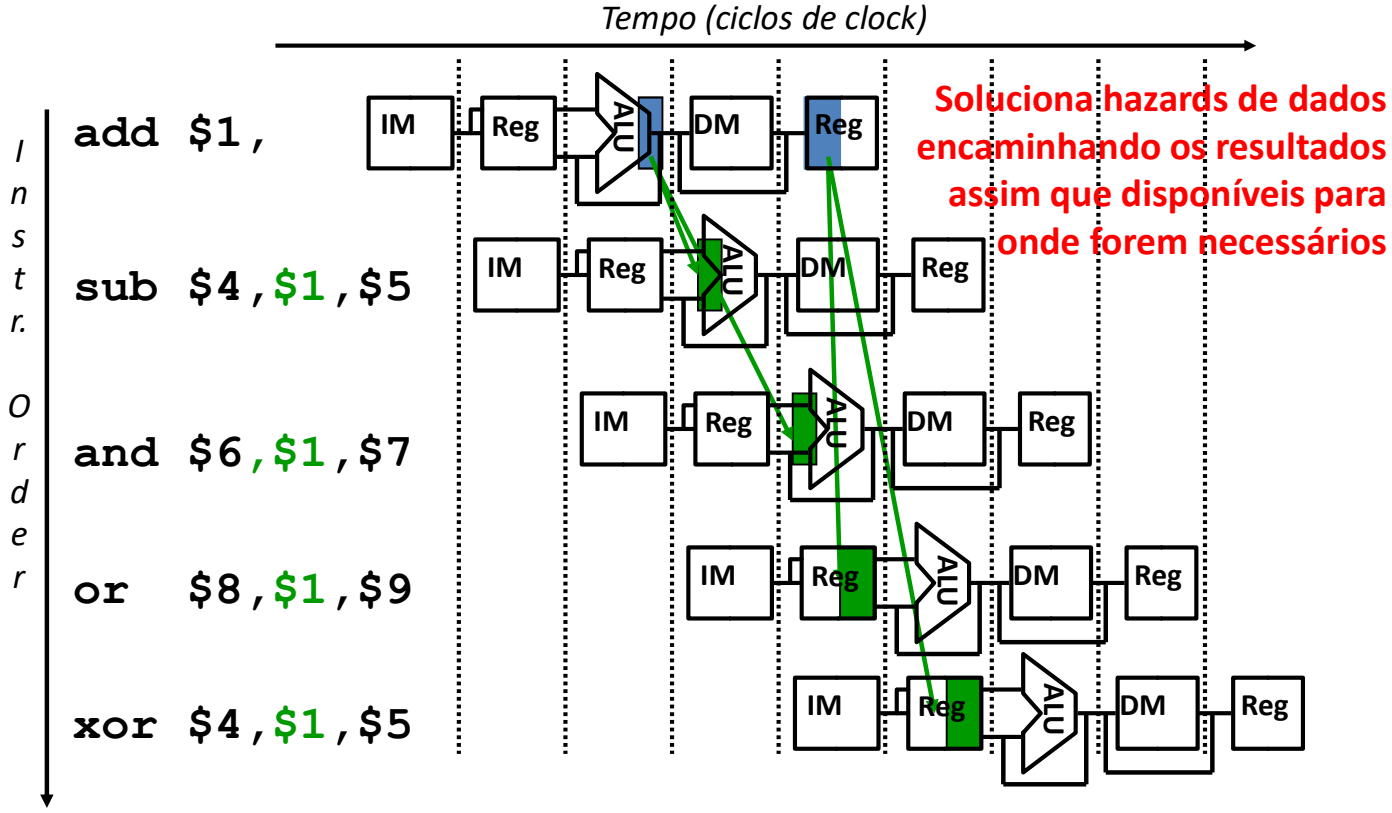
Caminho de dados com pipeline



Revisão – Hazard de dados – solução simples



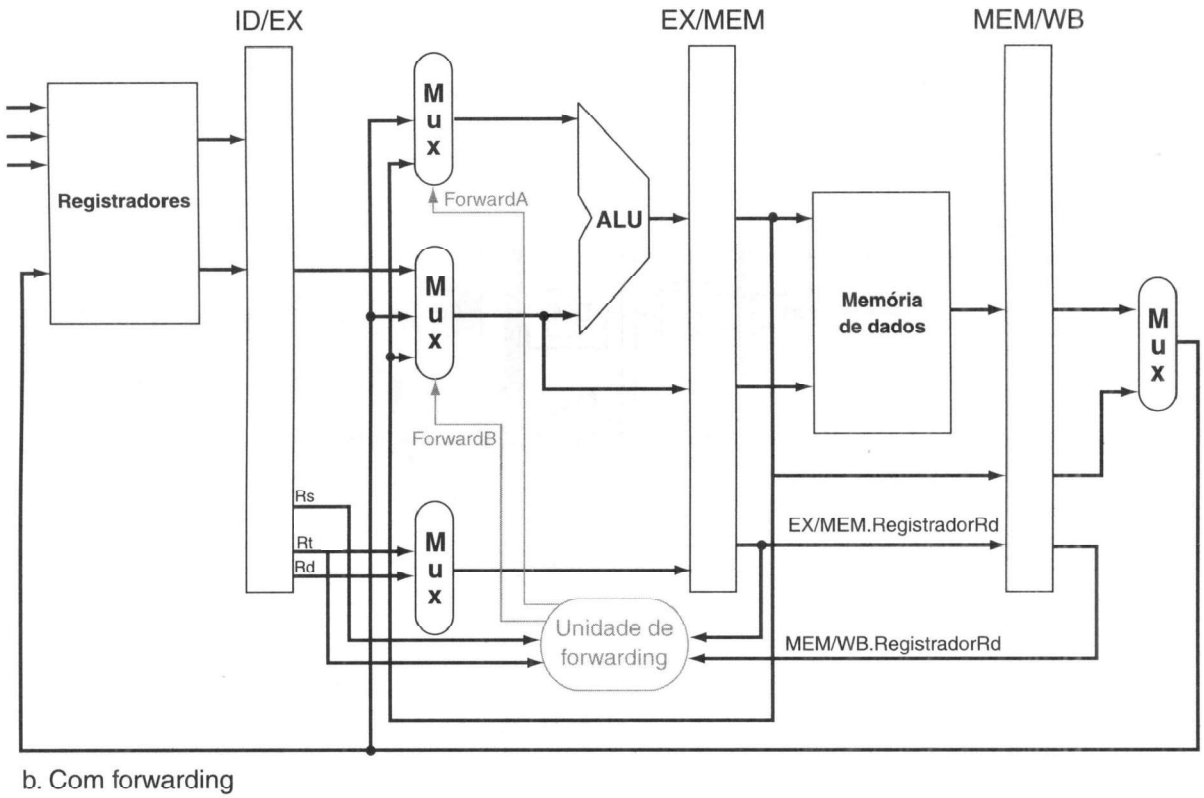
Revisão – Hazard de dados – encaminhamento



Solução por encaminhamento (forwarding)

- A implementação do encaminhamento necessita de uma forma de identificar o hazard.
- Deve-se coletar o resultado dos registradores de pipeline e realizar o encaminhamento para a ALU que necessita dos dados:
  - Para a ALU, as entradas vem de quaisquer registradores de pipeline ao invés do registrador ID/EX
  - Multiplexadores são adicionados nas entradas da ALU para selecionados os dados corretos
  - Adiciona-se um controle de hardware apropriado para controlar os novos multiplexadores
  - Outras unidades funcionais, como DM, necessitam de uma lógica similar
- Com *forwarding*, é possível atingir um CPI igual a 1 mesmo quando existirem dependências de dados

Caminho de dados após a inclusão do encaminhamento



b. Com forwarding

Caminho de dados após a inclusão do encaminhamento

Controle do MUX	Origem	Explicação
ForwardA = 00	ID/EX	O primeiro operando da ULA vem do banco de registradores.
ForwardA = 10	EX/MEM	O primeiro operando da ULA sofre forwarding do resultado anterior da ULA.
ForwardA = 01	MEM/WB	O primeiro operando da ULA sofre forwarding da memória de dados ou de um resultado anterior da ULA.
ForwardB = 00	ID/EX	O segundo operando da ULA vem do banco de registradores.
ForwardB = 10	EX/MEM	O segundo operando da ULA sofre forwarding do resultado anterior da ULA.
ForwardB = 01	MEM/WB	O segundo operando da ULA sofre forwaring da memória de dados ou de um resultado anterior da ULA.

Caminho de dados após a inclusão do encaminhamento

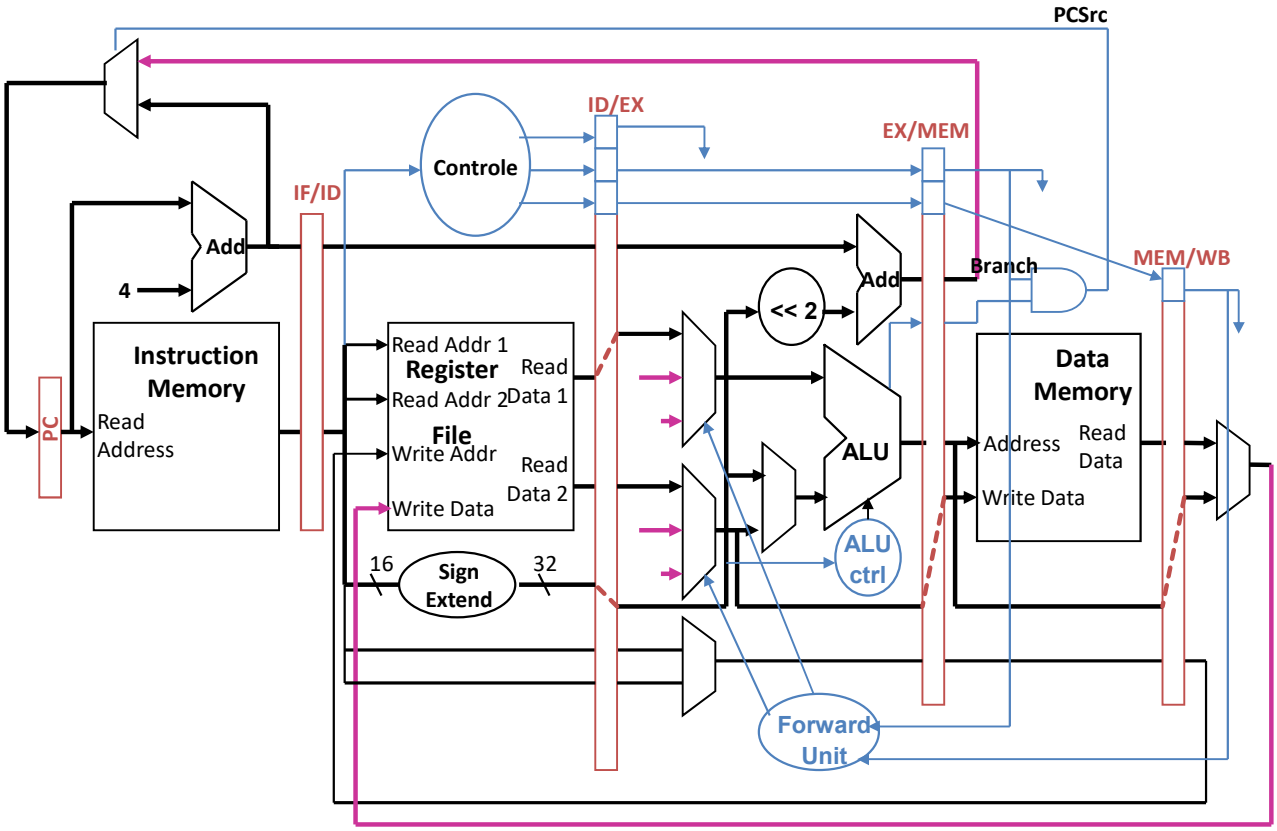
EX/MEM hazard:

```
if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd != 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
            ForwardA = 10
if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd != 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
            ForwardB = 10
```

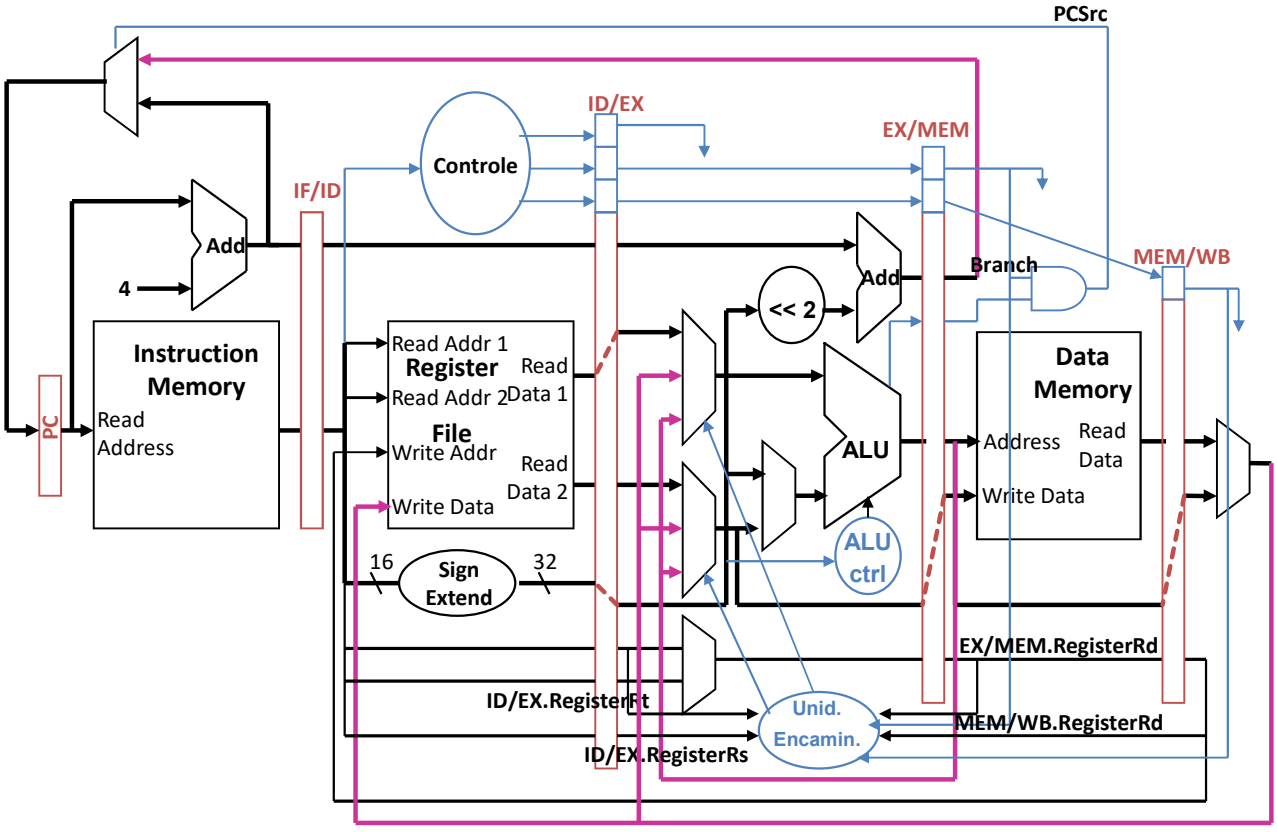
MEM/WB hazard:

```
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
            ForwardA = 01
if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd != 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
            ForwardB = 01
```

Caminho de dados - encaminhamento

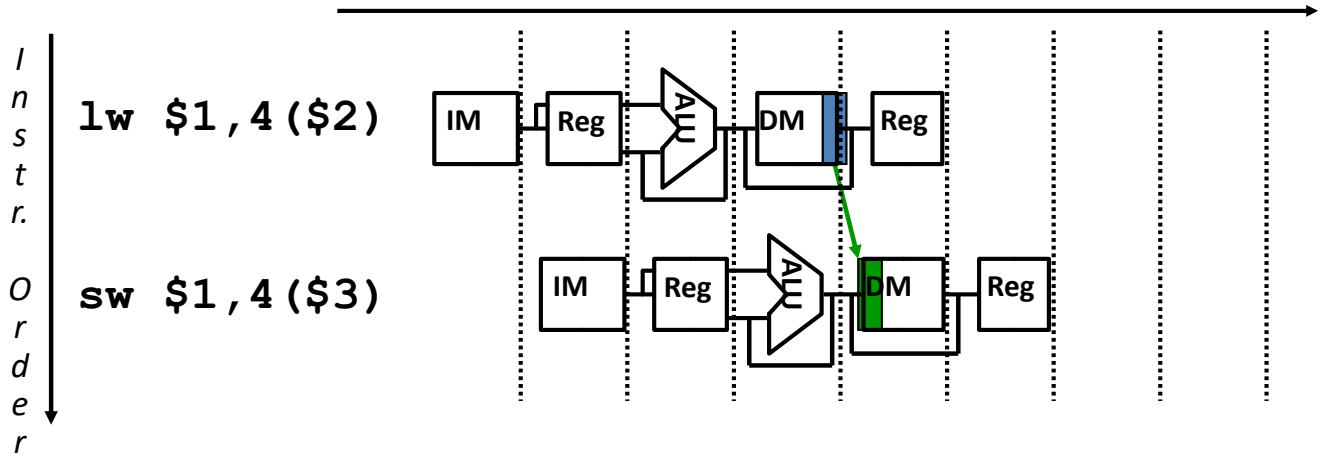


Caminho de dados - encaminhamento

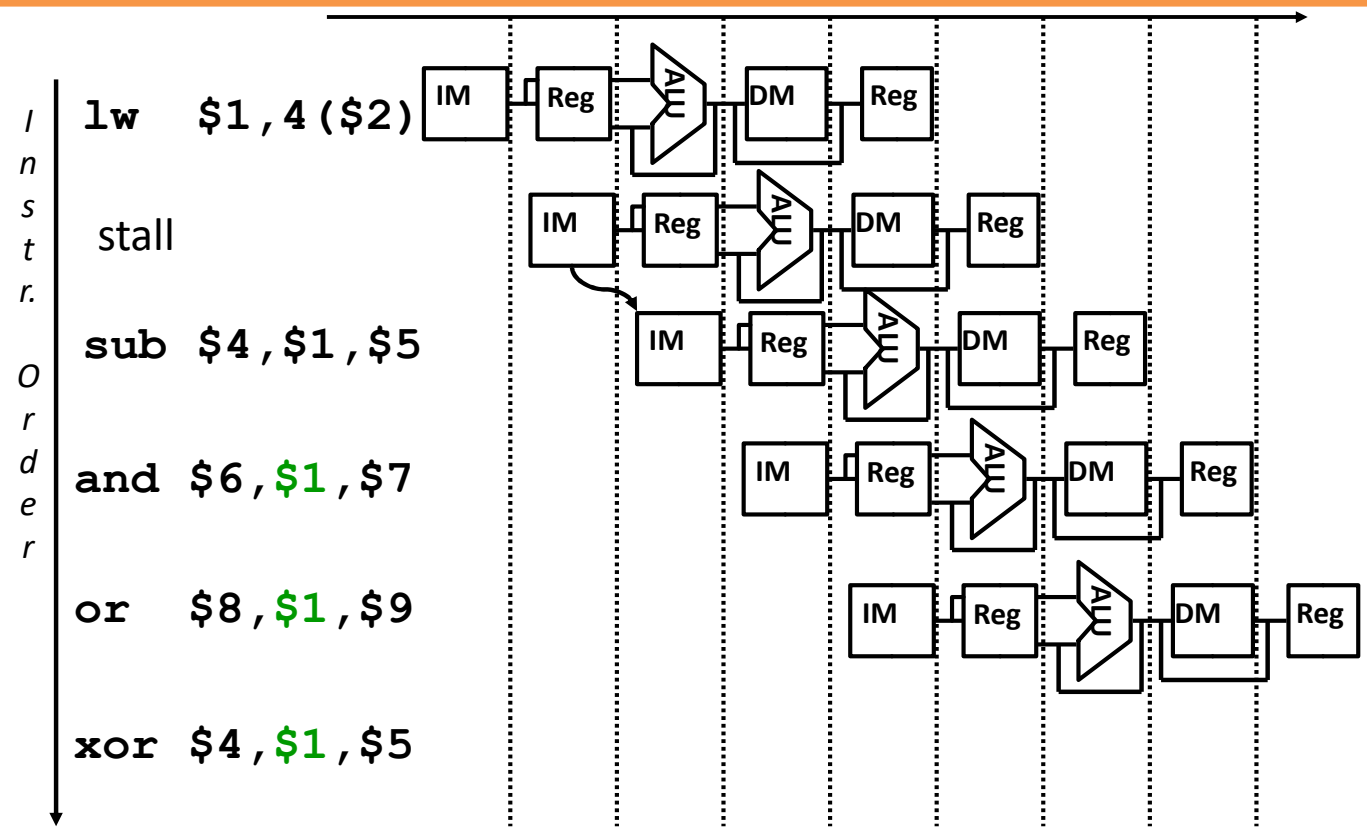


Acessos a memória

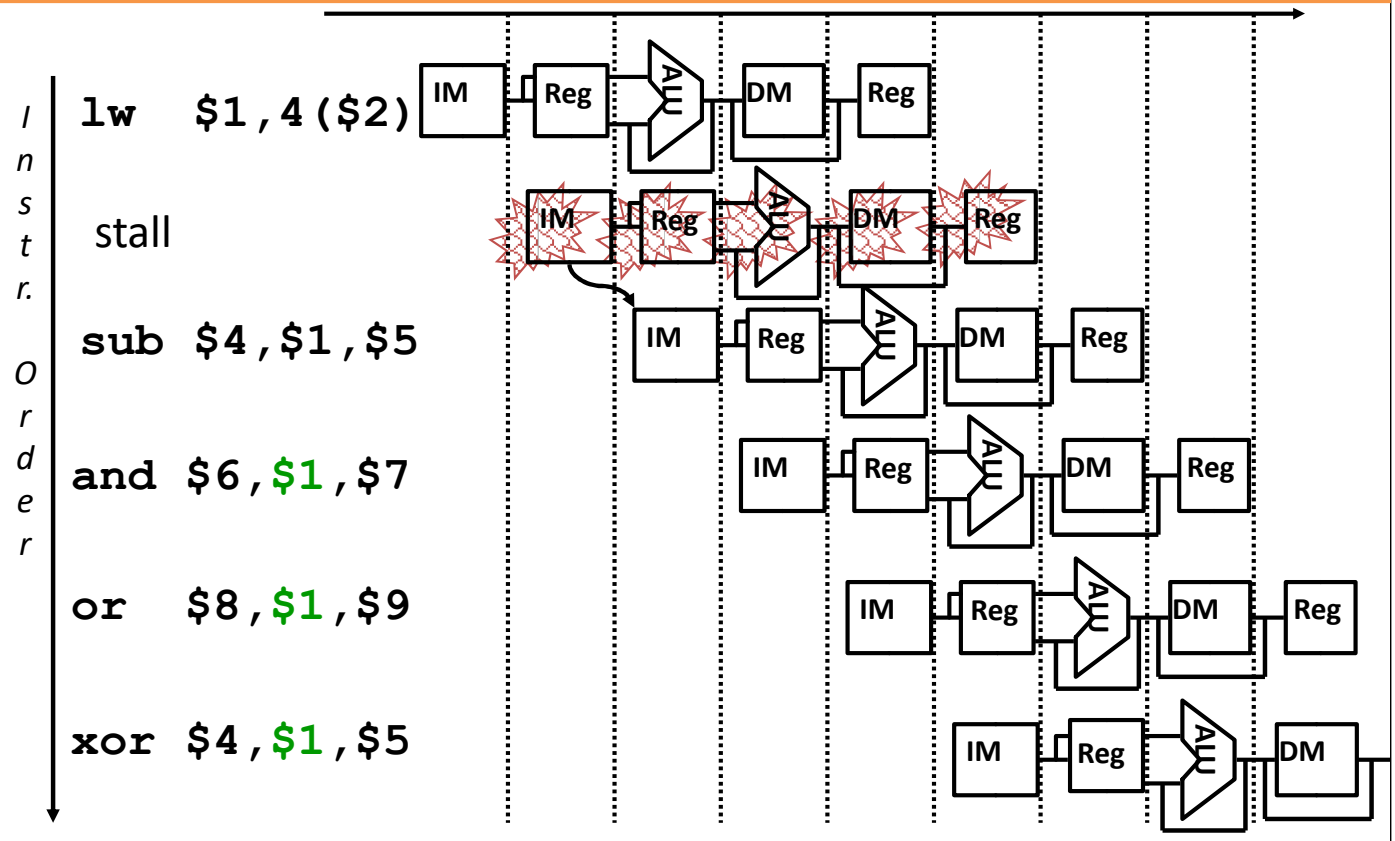
- Para loads imediatamente seguidos de stores, stalls podem ser evitados adicionando hardware de encaminhamento a partir do registrador MEM/WB para a entrada da memória de dados



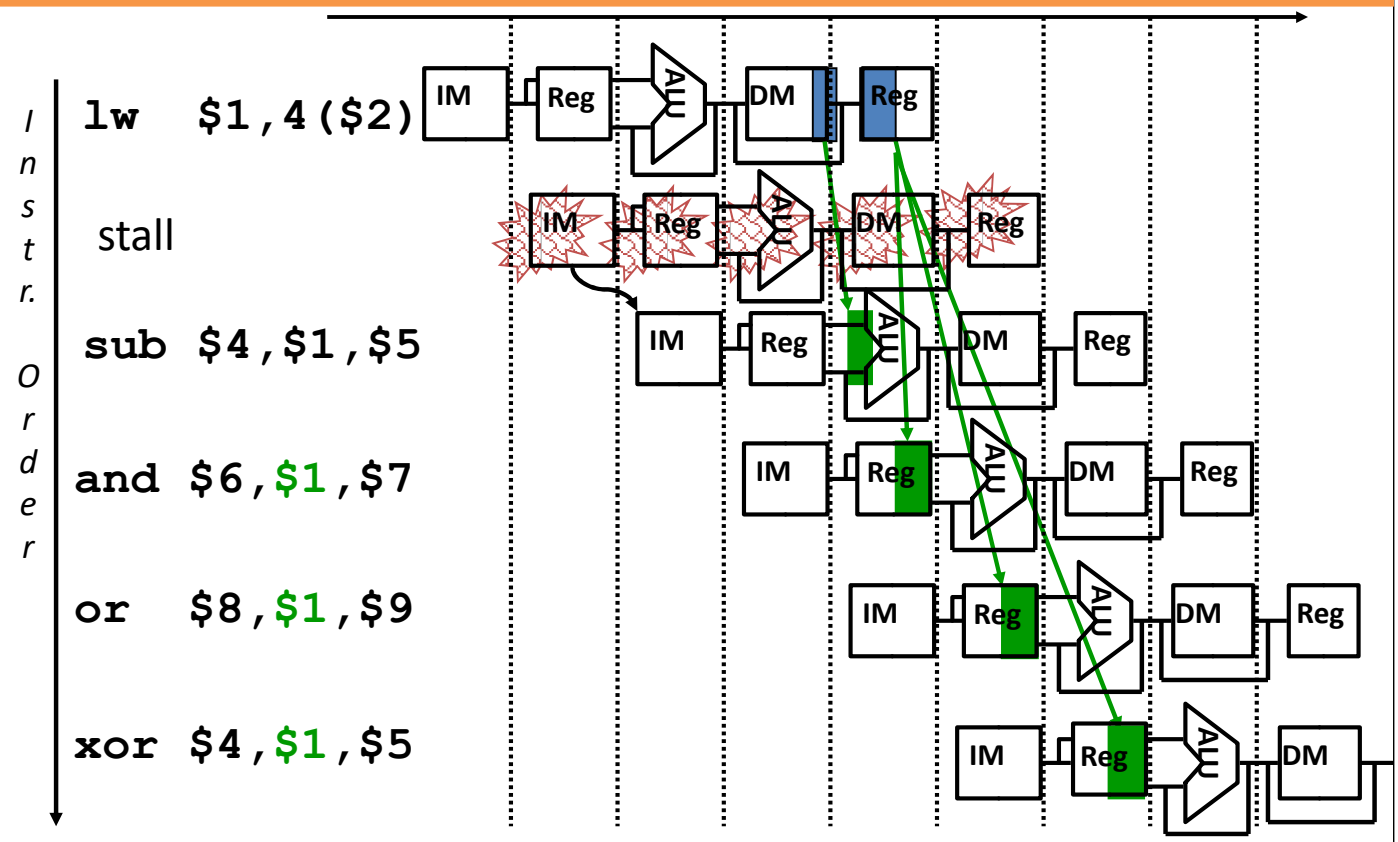
Load-use hazards



Load-use hazards



Load-use hazards



Load-use hazards

- Hardware de detecção de hazard no estágio ID:
  - Insere um stall entre a instrução load e o seu uso.
- Detecção de Hazard ID

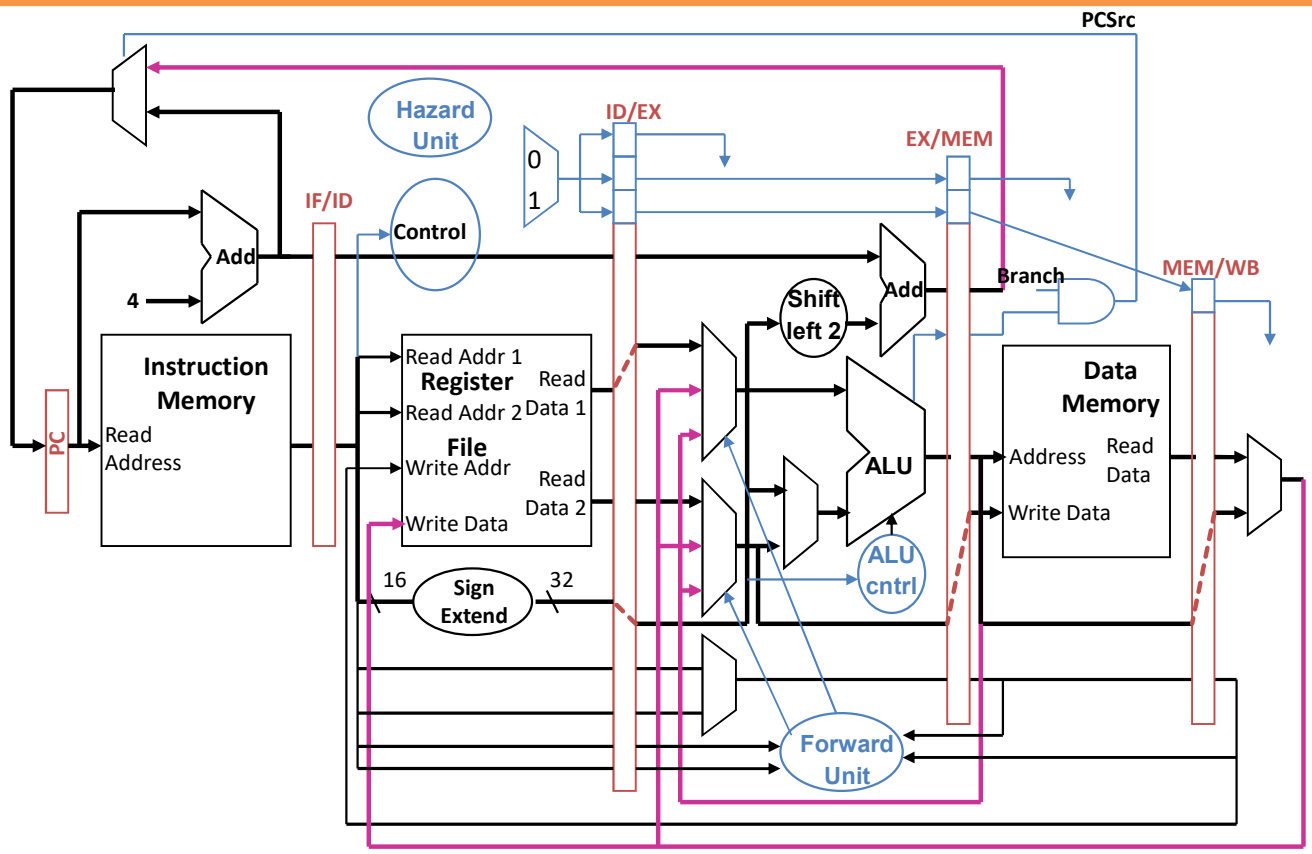
```
if (ID/EX.MemRead
    and ((ID/EX.RegisterRt = IF/ID.RegisterRs)
    or (ID/EX.RegisterRt = IF/ID.RegisterRt)))
    stall the pipeline
```

- A primeira linha testa se a instrução no estágio de execução é um lw.
- As duas linhas seguintes verificam se o registrador de destino do lw casa com qualquer registrador fonte da instrução que está no estágio ID (instrução load-use).
- Após uma bolha de um ciclo, a lógica de encaminhamento pode corrigir os hazards de dados restantes.

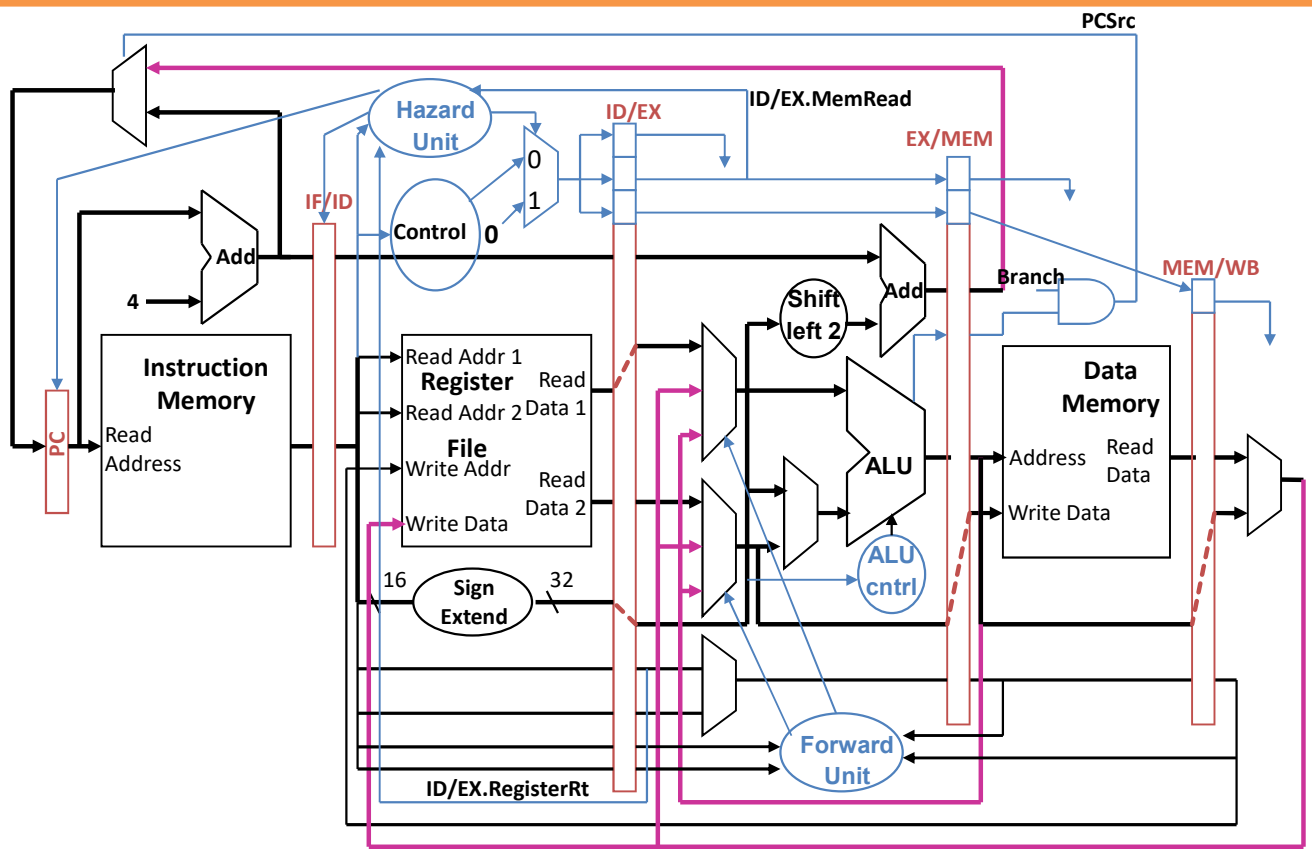
Load-use hazards

- A unidade de detecção de hazard controla a escrita para o PC e para os registradores IF/ID (PC.write e IF/ID.write)
  - Objetivo: Impedir que as instruções que estão nos estágios IF e ID caminhem pelo pipeline.
- Insere uma bolha entre a instrução lw e a instrução no estágio ID (isto é, insere uma instrução nop)
  - Os bits de controle nos registradores EX, MEM, WB são setados como 0
  - A instrução após a instrução lw continua sua execução no pipeline

Caminho de dados com detecção de hazards



Caminho de dados com detecção de hazards

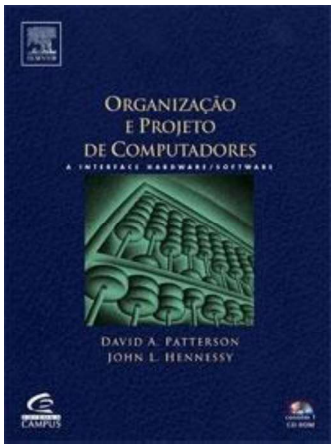


Bibliografia

1. PATTERSON, D.A; HENNESSY, J.L. **Organização e Projeto de Computadores: A Interface Hardware/Software**. 3a. Ed. Elsevier, 2005.
   - Capítulo ***5.***

2. Notas de aula do prof. Luciano J. Senger:
   - http://www.ljsenger.net/classroom.html

3. Notas de aula da Profa. Mary Jane Irwin
   - CSE 331 Computer Organization and Design
   - http://www.cse.psu.edu/research/mdl/mji/mjicourses



FIM

- FIM:
  - **Aula 12 –** Pipeline do MIPS 3: Solucionando hazards de dados
- Próxima aula:
  - **Aula 13 –** Pipeline do MIPS 4: Solucionando hazards de controle