Laboratório de Sistemas Embarcados e Distribuídos

Organização do RISC-V: Pipeline

Histórico de revisões

Revisão	Data	Responsável	Descrição
0.1	- X -	Prof. Cesar Zeferino	Primeira versão
0.2	03/2016	Prof. Cesar Zeferino	Revisão do modelo e atualização de conteúdo
0.3	06/2017	Prof. Cesar Zeferino	Pequenas correções
0.4	05/2020	Prof. Cesar Zeferino	Revisão geral
0.5	03/2023	Prof. Felski	Revisão da arquitetura para RISC-V

Observação: Este material foi produzido por pesquisadores do Laboratório de Sistemas Embarcados e Distribuídos (LEDS – Laboratory of Embedded and Distributed Systems) da Universidade do Vale do Itajaí e é destinado para uso em aulas ministradas por seus pesquisadores.

Introdução

- **Objetivo**
  - Conhecer aspectos do projeto de um processador para aumentar o desempenho por meio do uso da técnica de processamento em pipeline
- **Conteúdo**
  - Ganho de desempenho com o pipeline
  - O caminho de dados
  - O controle
  - Representação do pipeline
  - O pipeline ideal
  - Tratamento de exceções

Introdução

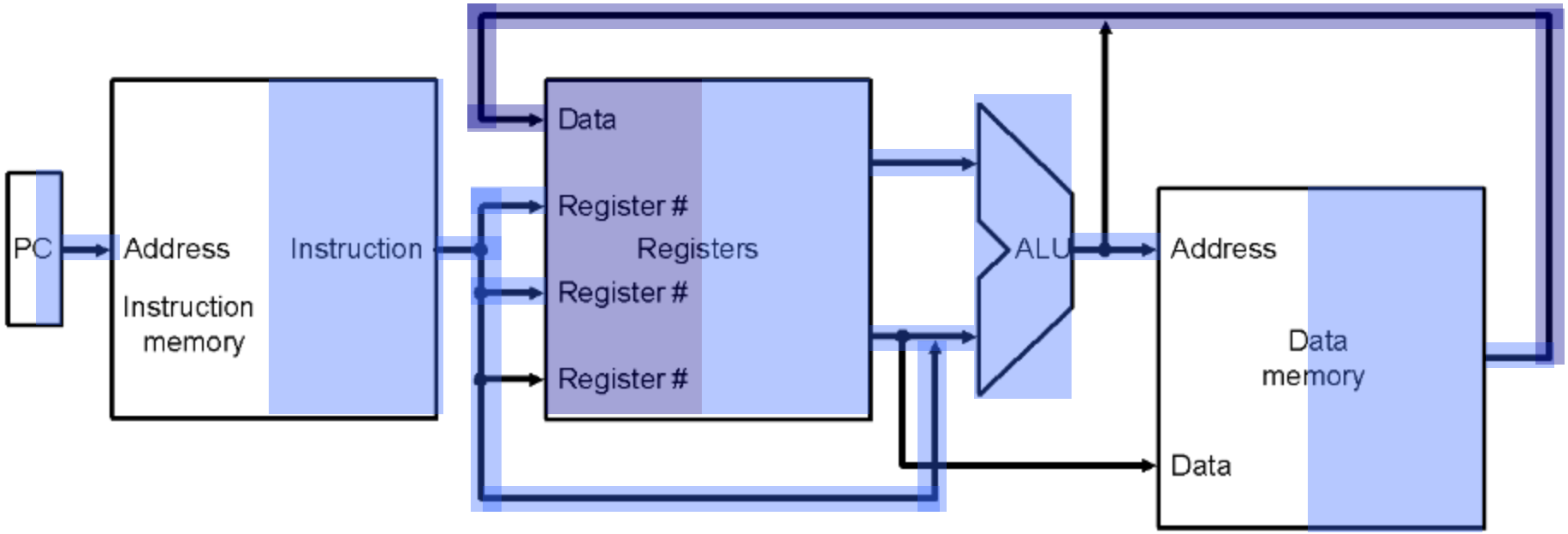
Bibliografia

- PATTERSON, David A.; HENNESSY, John L. O Processador. *In*: _____. **Organização e projeto de computadores**: a interface hardware/software. 4. ed. Rio de Janeiro: Campus, 2014. cap. 4.

Introdução

- No RISC-V monociclo, o período do relógio é definido pela instrução que utiliza mais recursos do processador (a mais lenta) – ou seja *load word*

- Isso resulta em um período de relógio (T_{clk}) grande

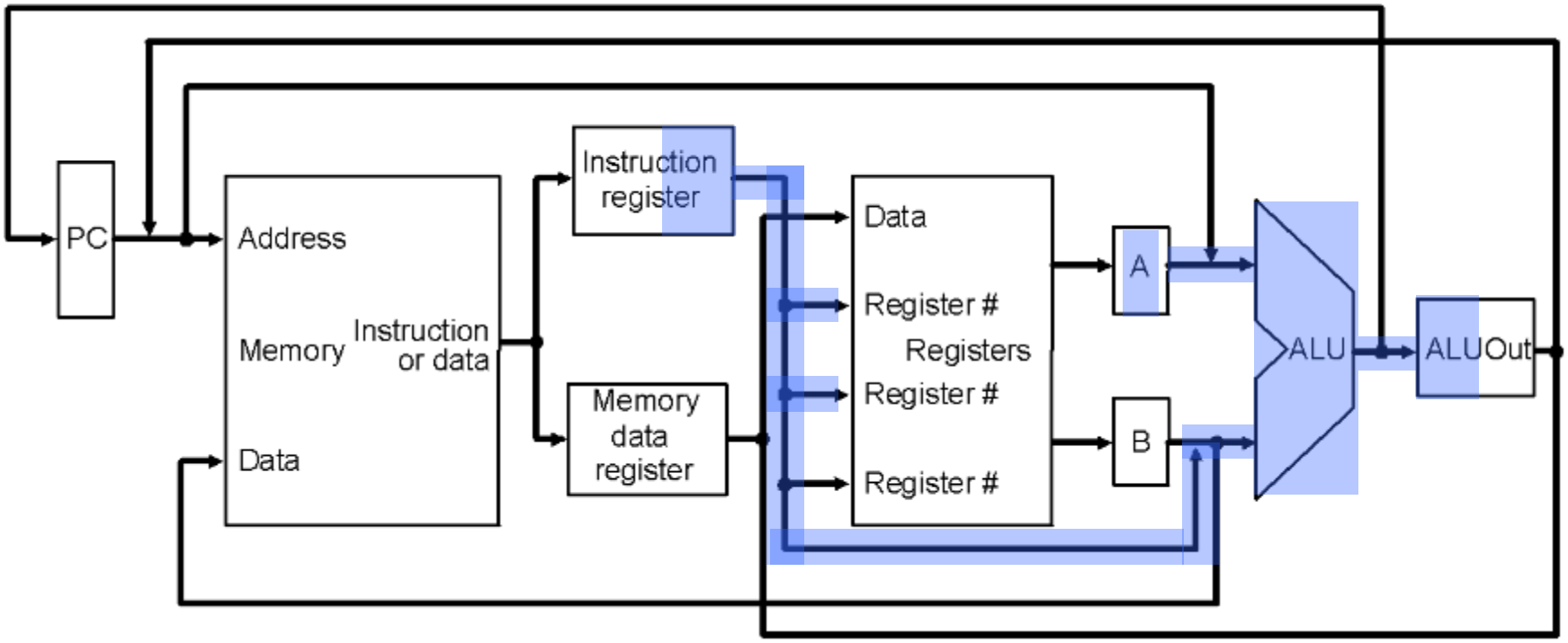


IF ▸ ID ▸ EXE ▸ MEM ▸ WB

Introdução

- **No RISC-V multiciclo, o período de relógio é definido pela etapa mais lenta do ciclo de instrução ($T_{clk\_multiciclo} < T_{clk\_monociclo}$)**

- **Quando uma etapa está sendo executada, apenas os recursos necessários são usados e os demais ficam ociosos**
  - Exemplo: Recursos ocupados durante a etapa de execução

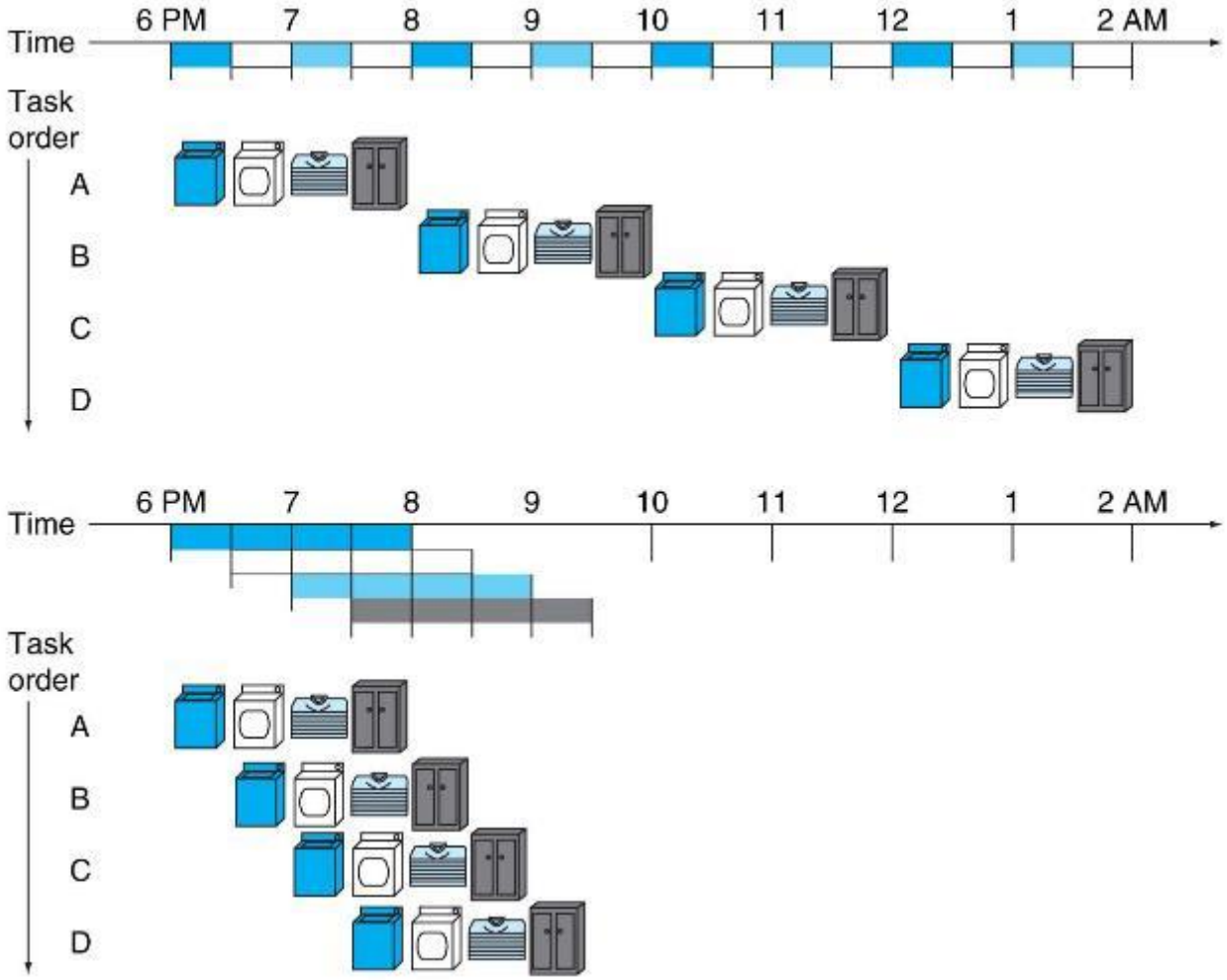


IF	ID	EXE	MEM	WB

Introdução

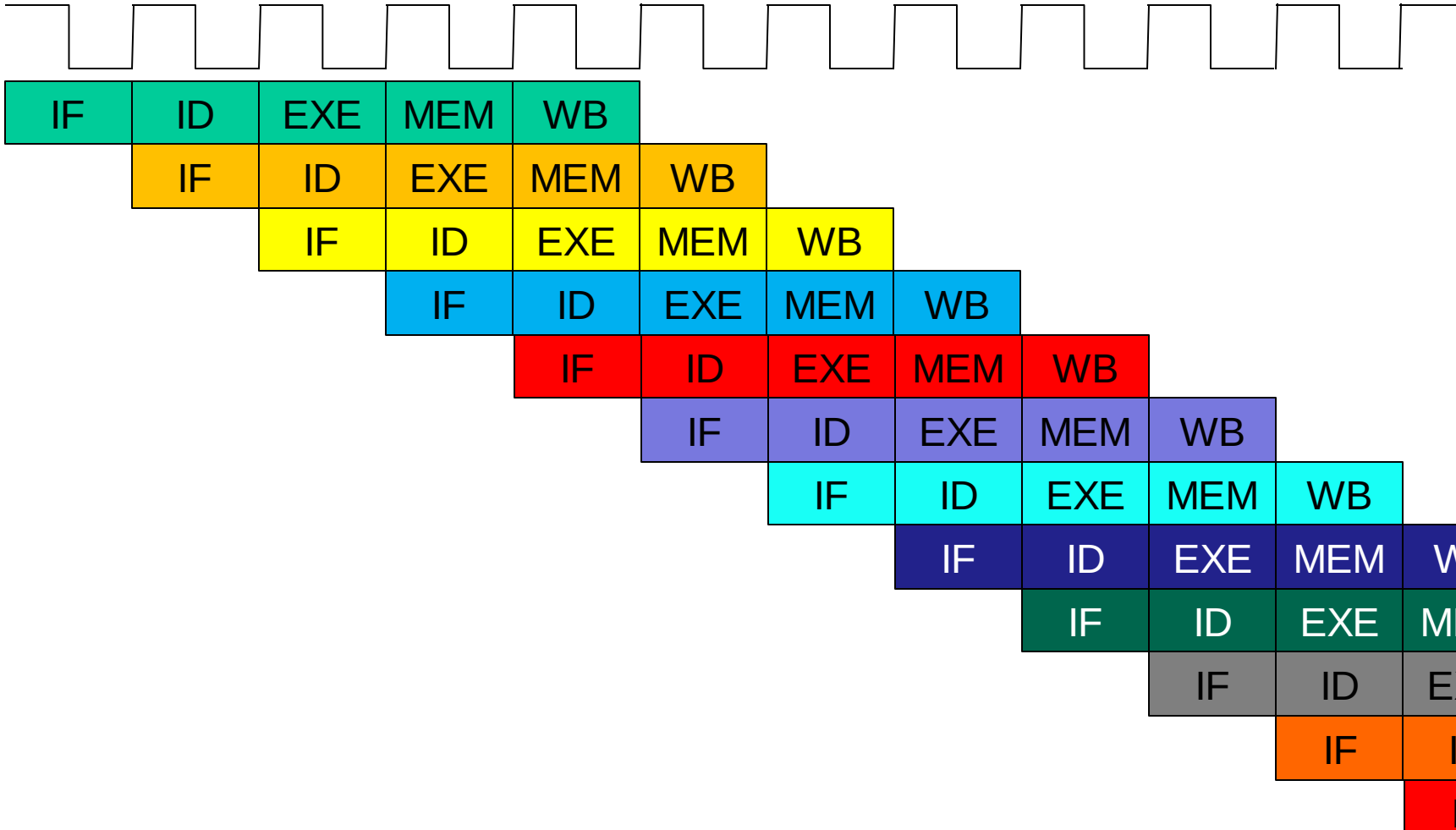
- **Os recursos não utilizados poderiam ser empregados para executar outras instruções para aumentar o desempenho**

- **Analogia**
  - Lavar lotes de roupa suja em uma lavanderia
  - Multiciclo: 0,5 lote por hora
  - Pipeline:
    - 1o lote: 2 horas
    - Próximos: 0,5 h
    - 2 lotes por hora



Introdução

- **Pipeline e instruções**
  - 5 ciclos para a primeira instrução
  - As demais ficam prontas em 1 ciclo



Ganho de desempenho com o pipeline

- **Premissas**
  - Tempos das operações básicas (estimativa para uma dada tecnologia, desconsiderando atrasos nas conexões)
    - Acesso à memória: 200 ps (etapas IF e MEM)
    - Acesso a registrador: 100 ps (etapas ID e WB)
    - Cálculo na ULA: 200 ps (etapa EXE)
  -

- **Período de relógio de cada organização**
  - $\mathbf{T_{clk\_monociclo}} = T_{IF} + T_{ID} + T_{EXE} + T_{MEM} + T_{WB} = \mathbf{800\ ps}$
  - $\mathbf{T_{clk\_multiciclo}} = max(T_{IF}, T_{ID}, T_{EXE}, T_{MEM}, T_{WB}) = \mathbf{200\ ps}$
  - $\mathbf{T_{clk\_pipeline}} = max(T_{IF}, T_{ID}, T_{EXE}, T_{MEM}, T_{WB}) = \mathbf{200\ ps}$

Ganho de desempenho com o pipeline

- **Tempo para executar um programa com I instruções**

$$t_{CPU} = I \times CPI \times T_{clk}$$

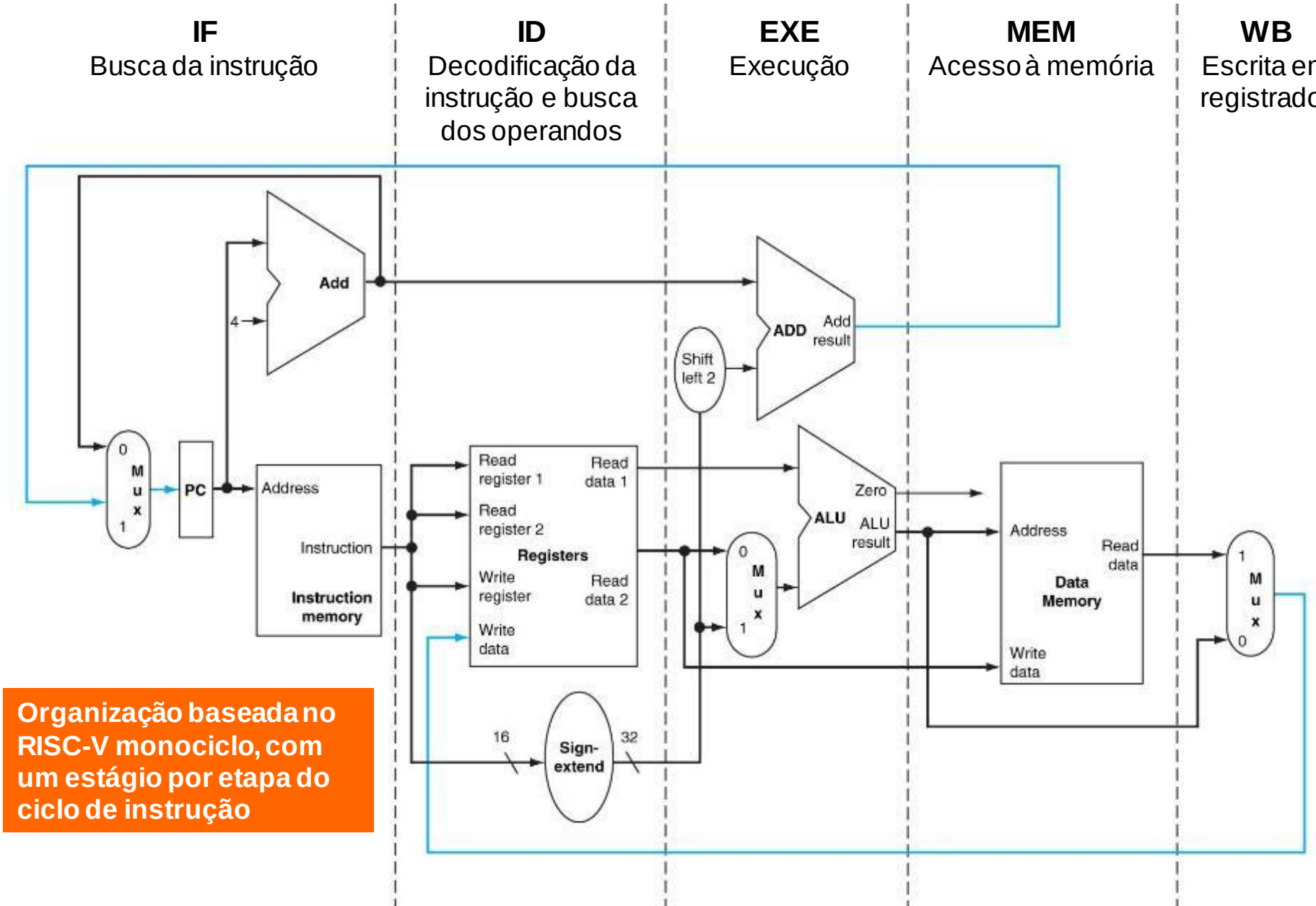
- **CPI – Ciclos por Instrução**
  - $CPI_{monociclo} = 1$
  - $CPI_{multiciclo} = 4$ em média, pois varia com o programa
  - $CPI_{pipeline} = [5 + 1.(I - 1)] / I$ ou seja: 5 para a primeira instrução, 1 para as demais instruções

Ganho de desempenho com o pipeline

- **Tempo para executar um programa com 1 bilhão de instruções**
  - $t_{monociclo} = 10^9 \times 1 \times (800 \times 10^{-12}) =$ **0,8 s**
  - $t_{multiciclo} = 10^9 \times 4 \times (200 \times 10^{-12}) =$ **0,8 s**
  - $t_{pipeline} = 10^9 \times \{ [5 + 1.(10^9 - 1)] / 10^9 \} \times (200 \times 10^{-12})$
    $= [5 + (10^9 - 1)] \times (200 \times 10^{-12}) =$ **0,2 s**

O processador em pipeline executa o programa 4 vezes mais rapidamente

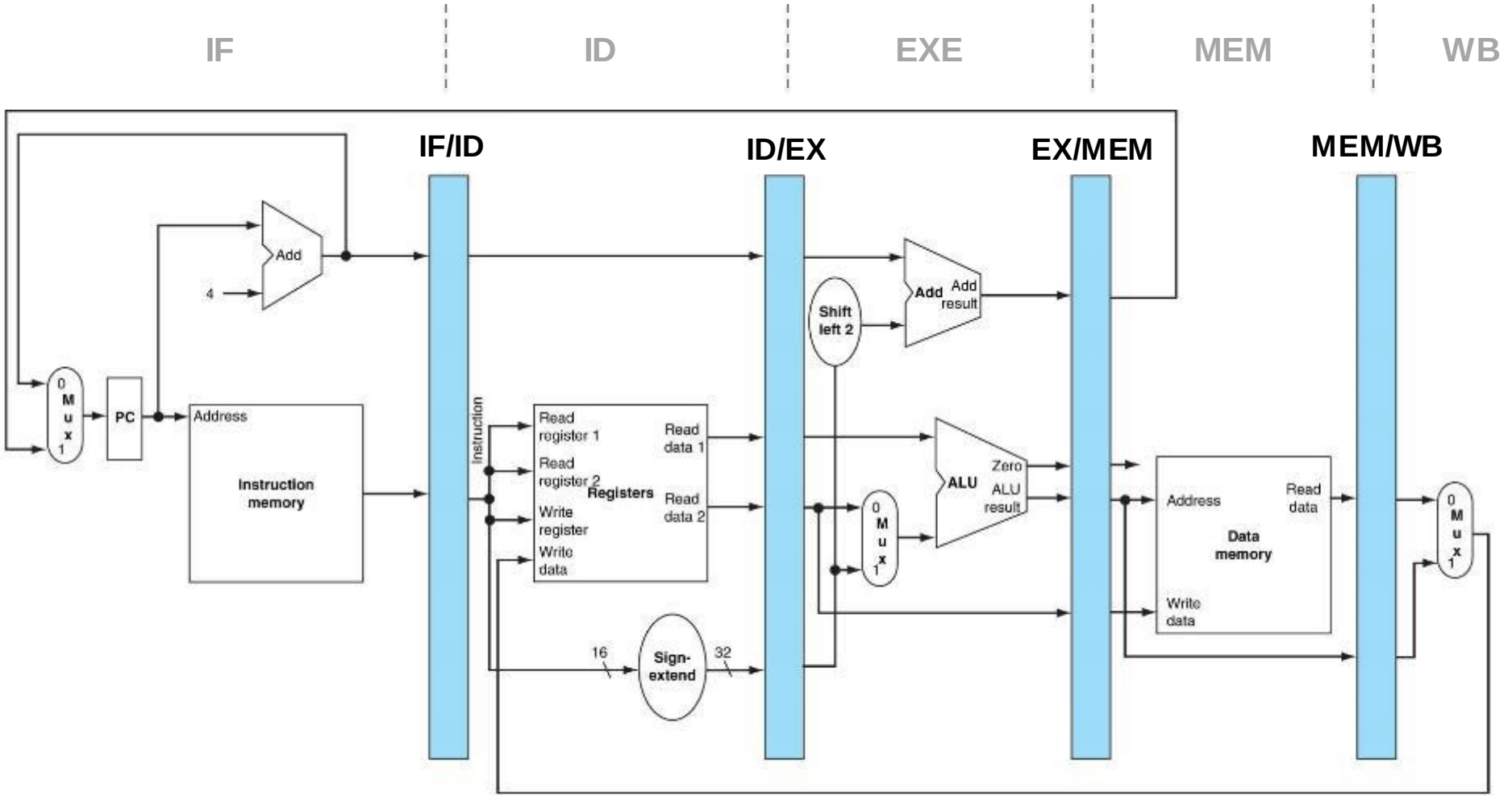
O caminho de dados



Organização baseada no RISC-V monociclo, com um estágio por etapa do ciclo de instrução

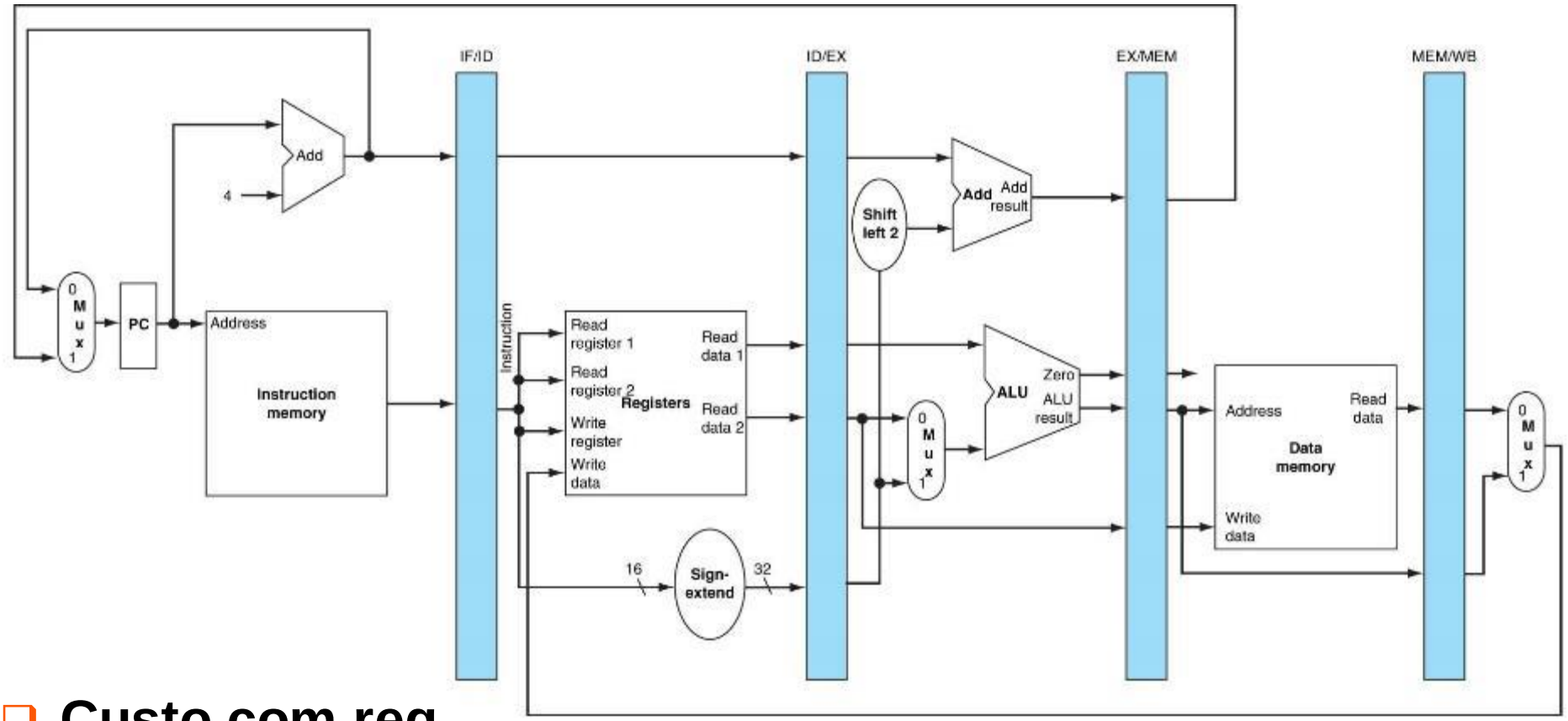
O caminho de dados

Registradores do pipeline



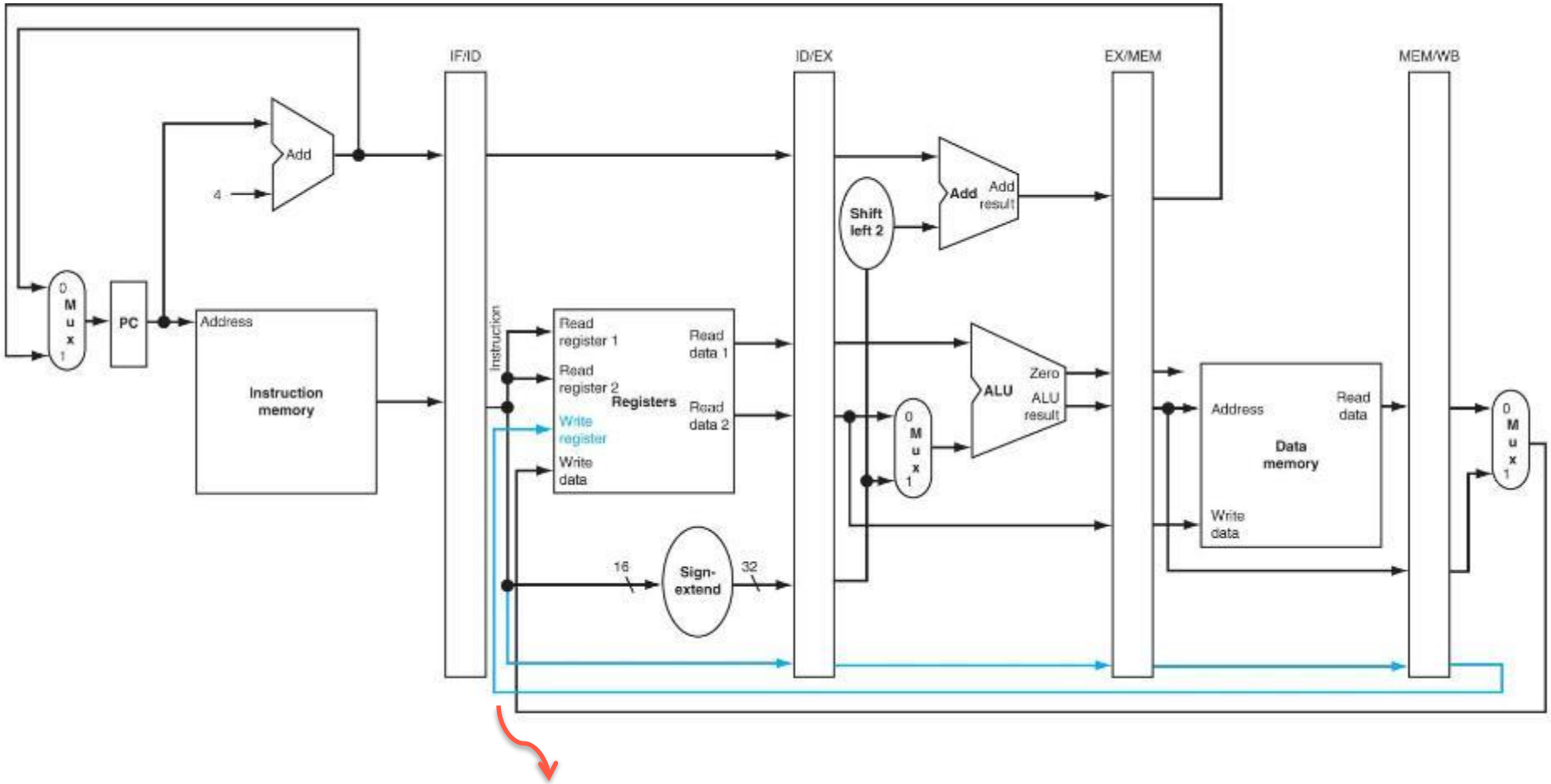
RISC-V pipeline = RISC-V monociclo + registradores do RISC-V multiciclo

O caminho de dados



- **Custo com reg.**
  - IF/ID = 32 + 32 = 64 FFs
  - ID/EX = 32 + 32 + 32 + 32 = 128 FFs
  - EX/MEM = 32 + 32 + 32 + 1 = 97 FFs
  - MEM/WB= 32 + 32 = 64 FFs

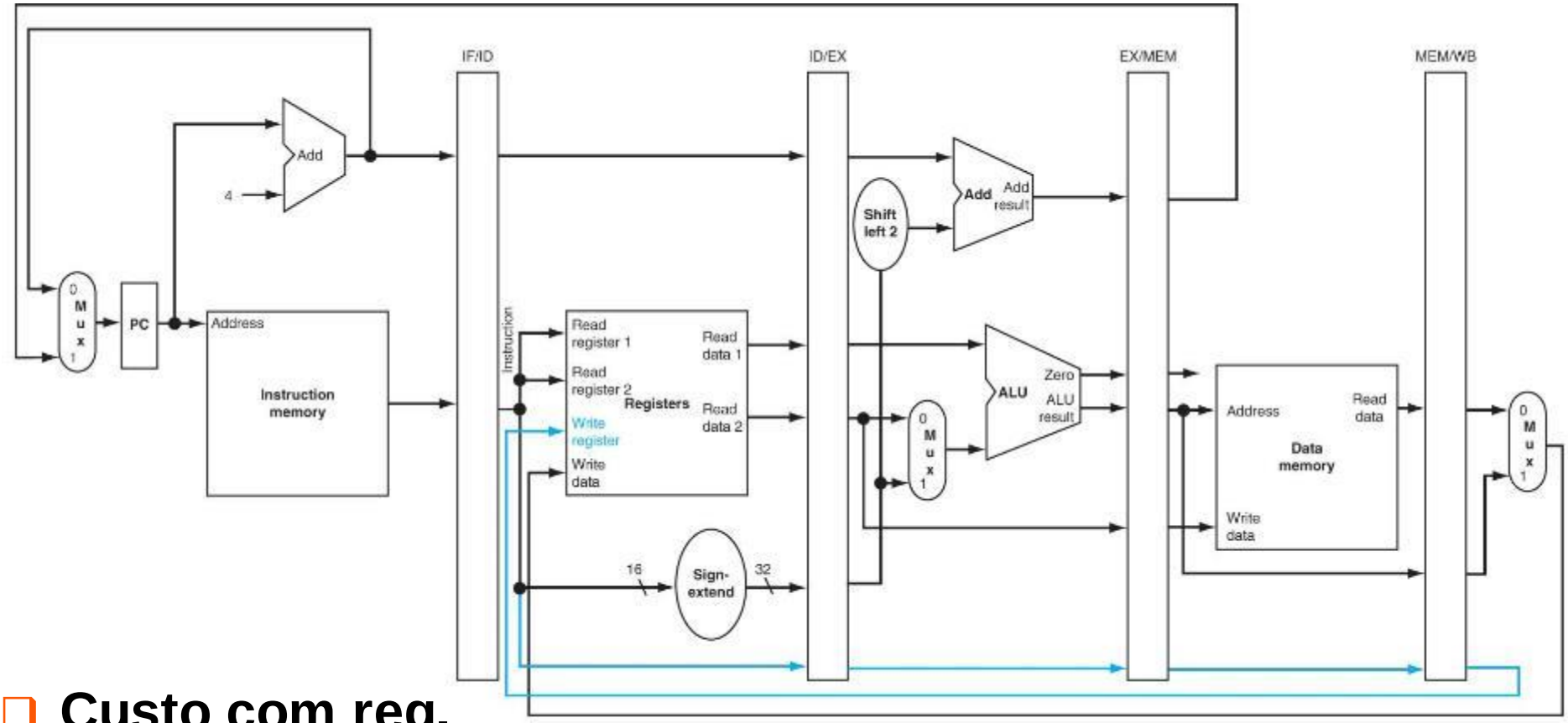
O caminho de dados



Propagação do endereço de escrita no registrador destino

(+ 5 FF nos registradores ID/EX, EX/MEM, MEM/WB)

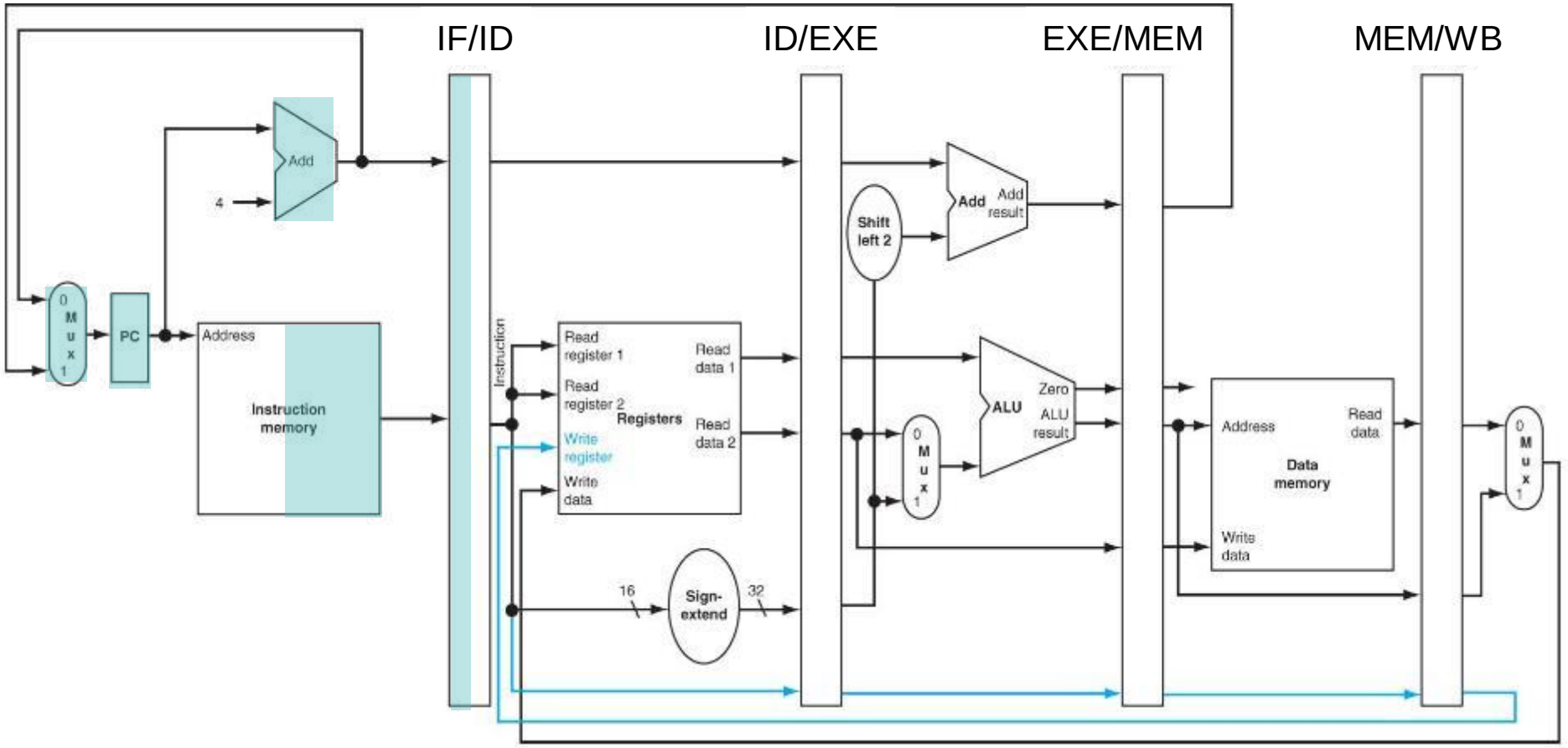
O caminho de dados



- **Custo com reg.**
  - IF/ID = 64 FFs
  - ID/EX = 128 + 5 = 133 FFs
  - EX/MEM = 97 + 5 = 102 FFs
  - MEM/WB= 64 + 5 = 69 FFs

Execução da instrução *lw*

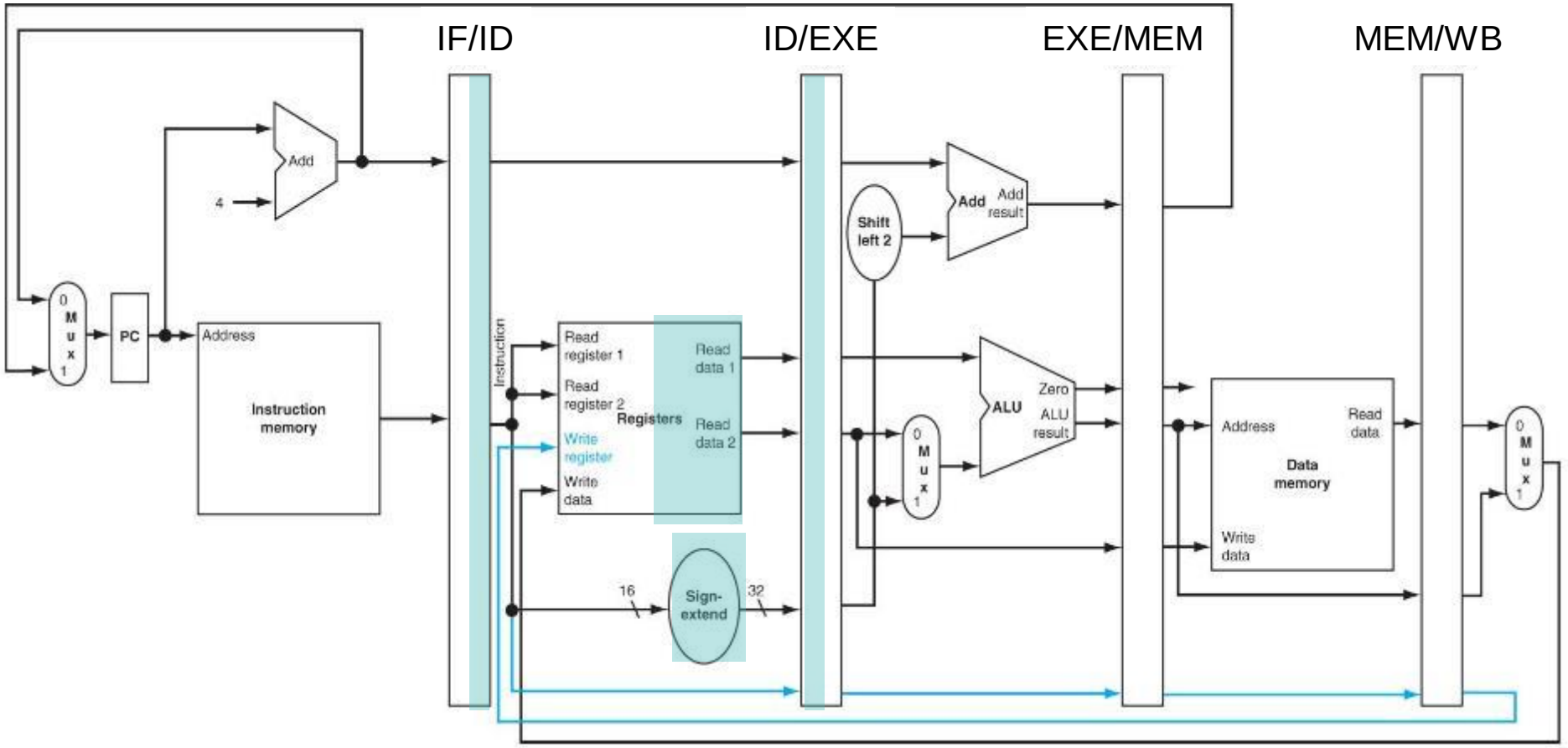
- **IF: Busca da instrução**



IF	ID	EXE	MEM	WB

Execução da instrução *lw*

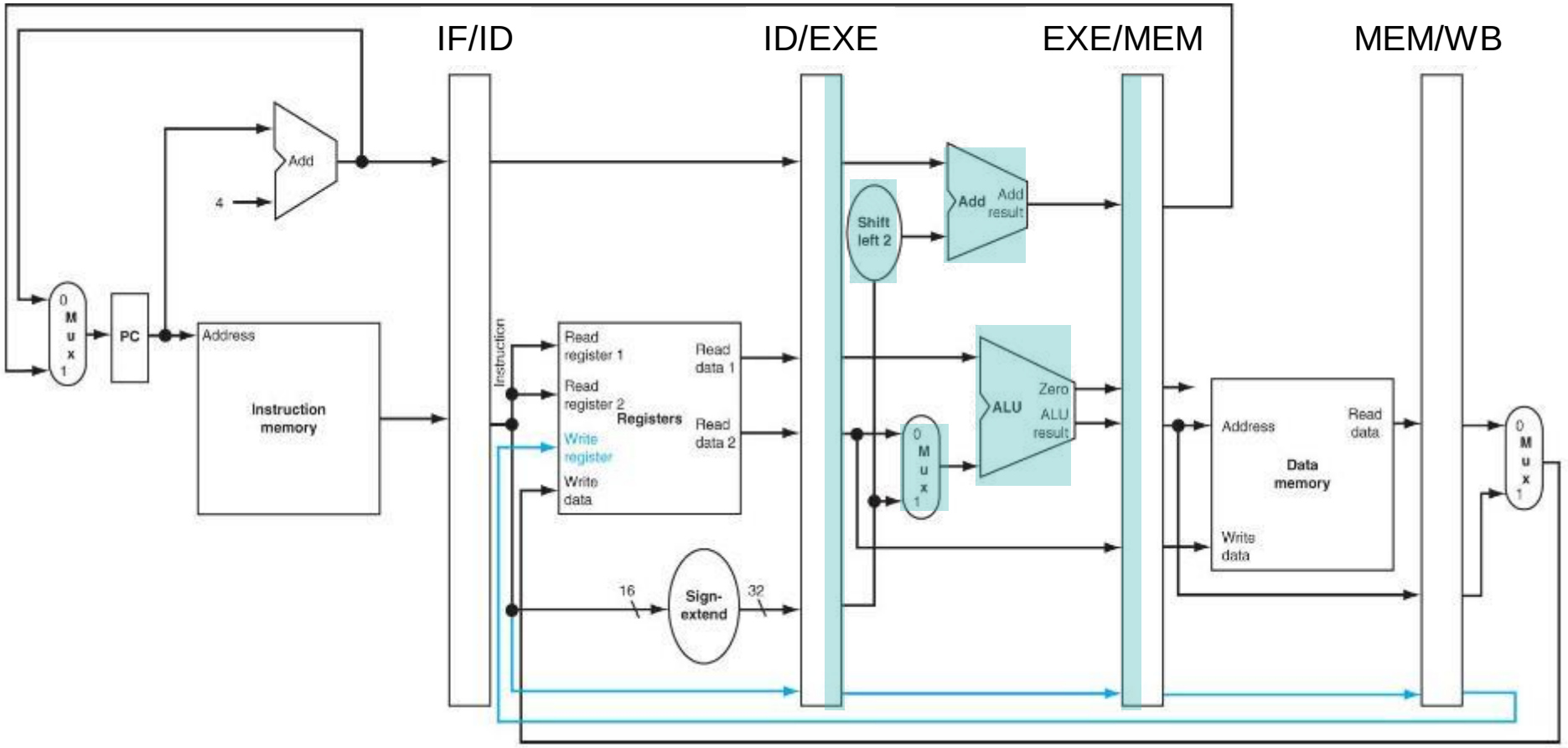
- **ID: Decodificação da instrução e busca dos operandos**



IF	ID	EXE	MEM	WB

Execução da instrução *lw*

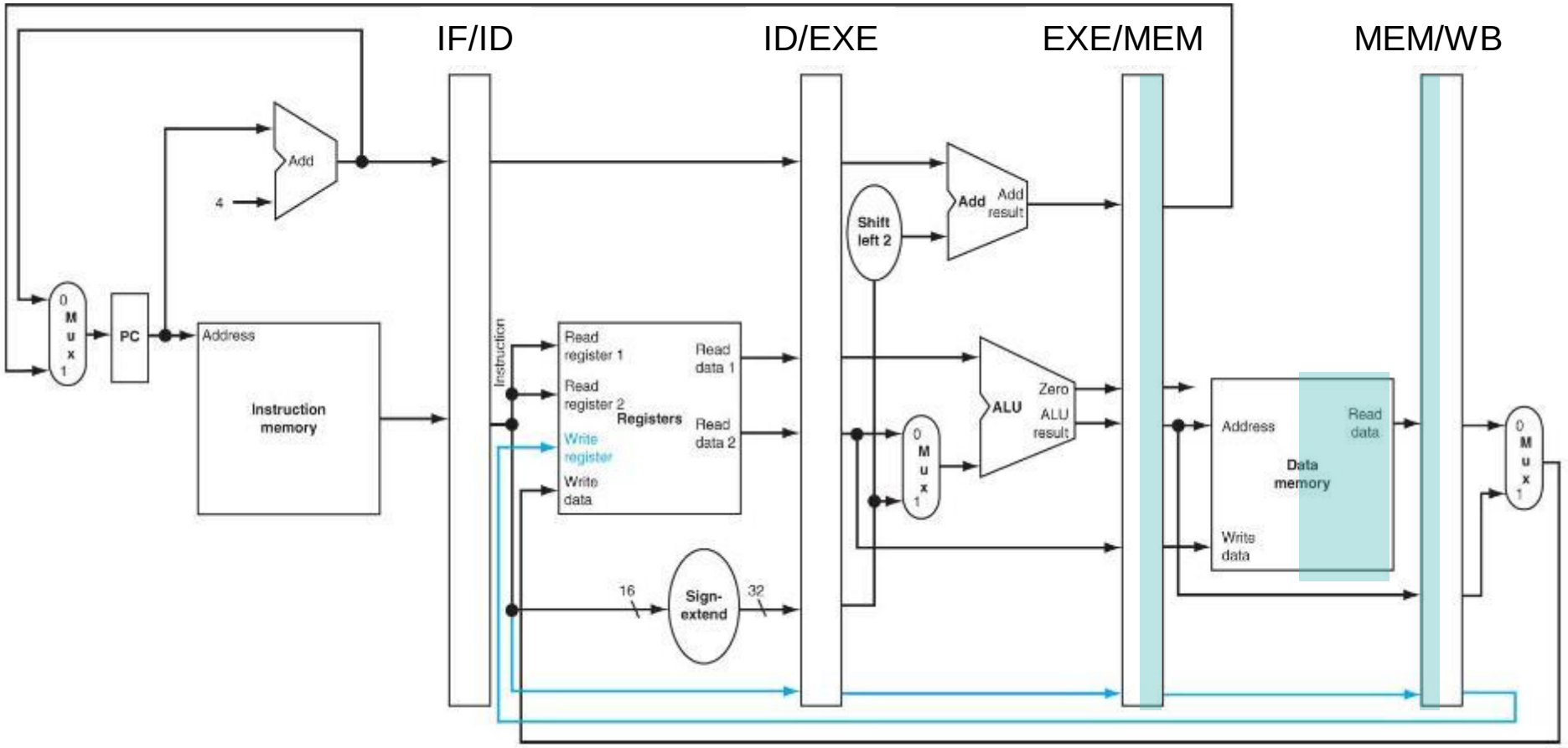
- **EXE: Cálculo do endereço efetivo**



IF	ID	EXE	MEM	WB

Execução da instrução *lw*

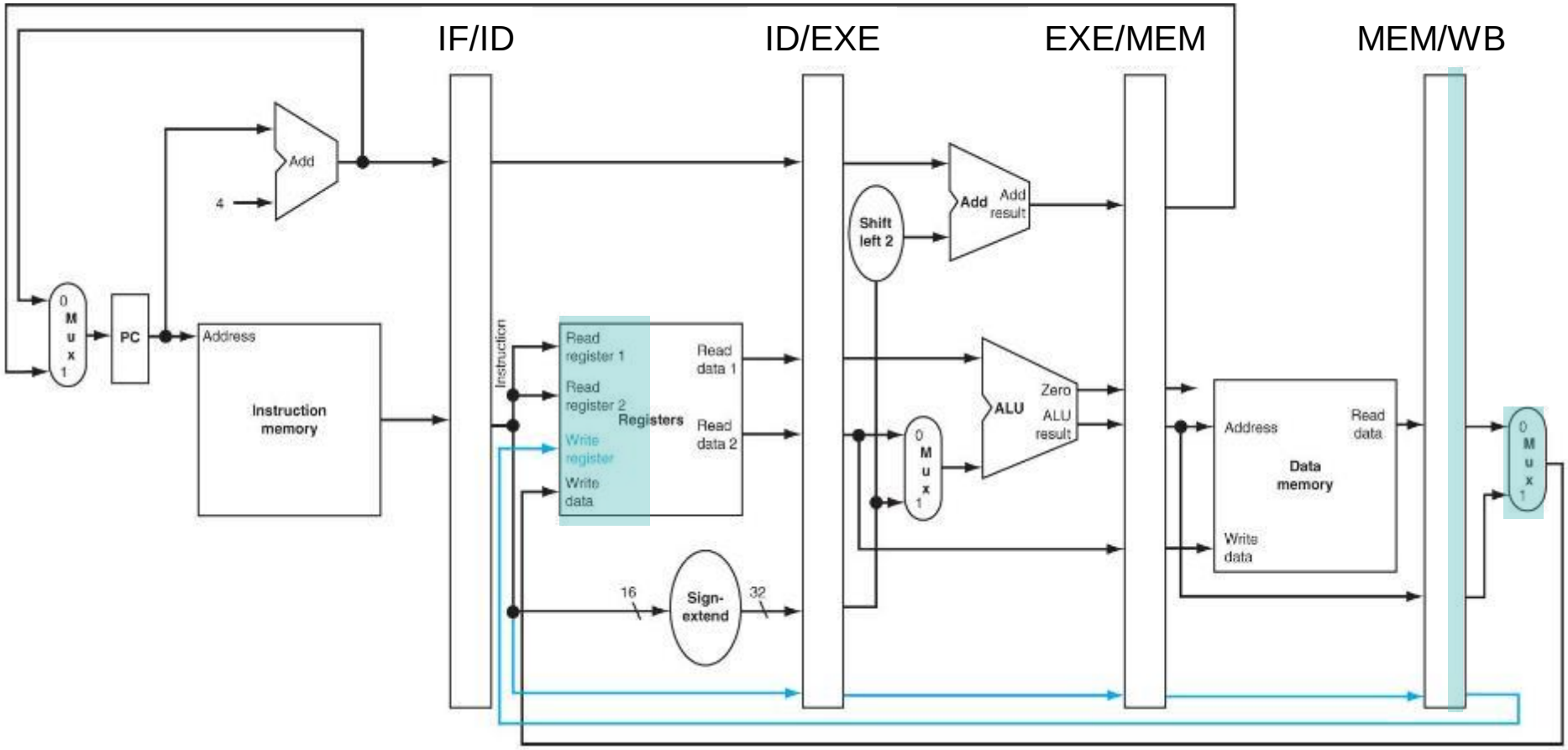
- **MEM: Acesso à memória**



IF	ID	EXE	MEM	WB

Execução da instrução *lw*

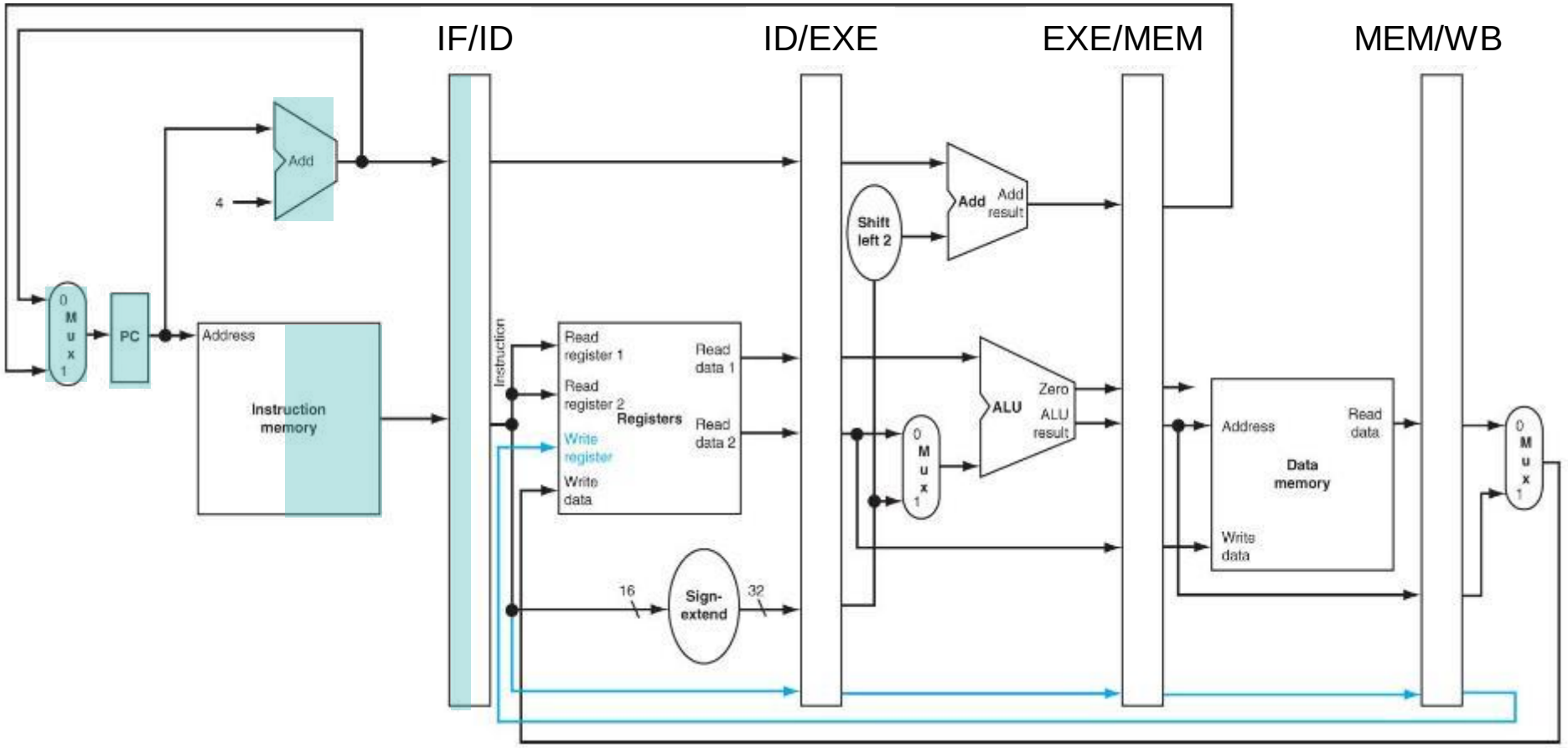
- **WB: Escrita no registrador**



IF	ID	EXE	MEM	WB

Execução da instrução *sw*

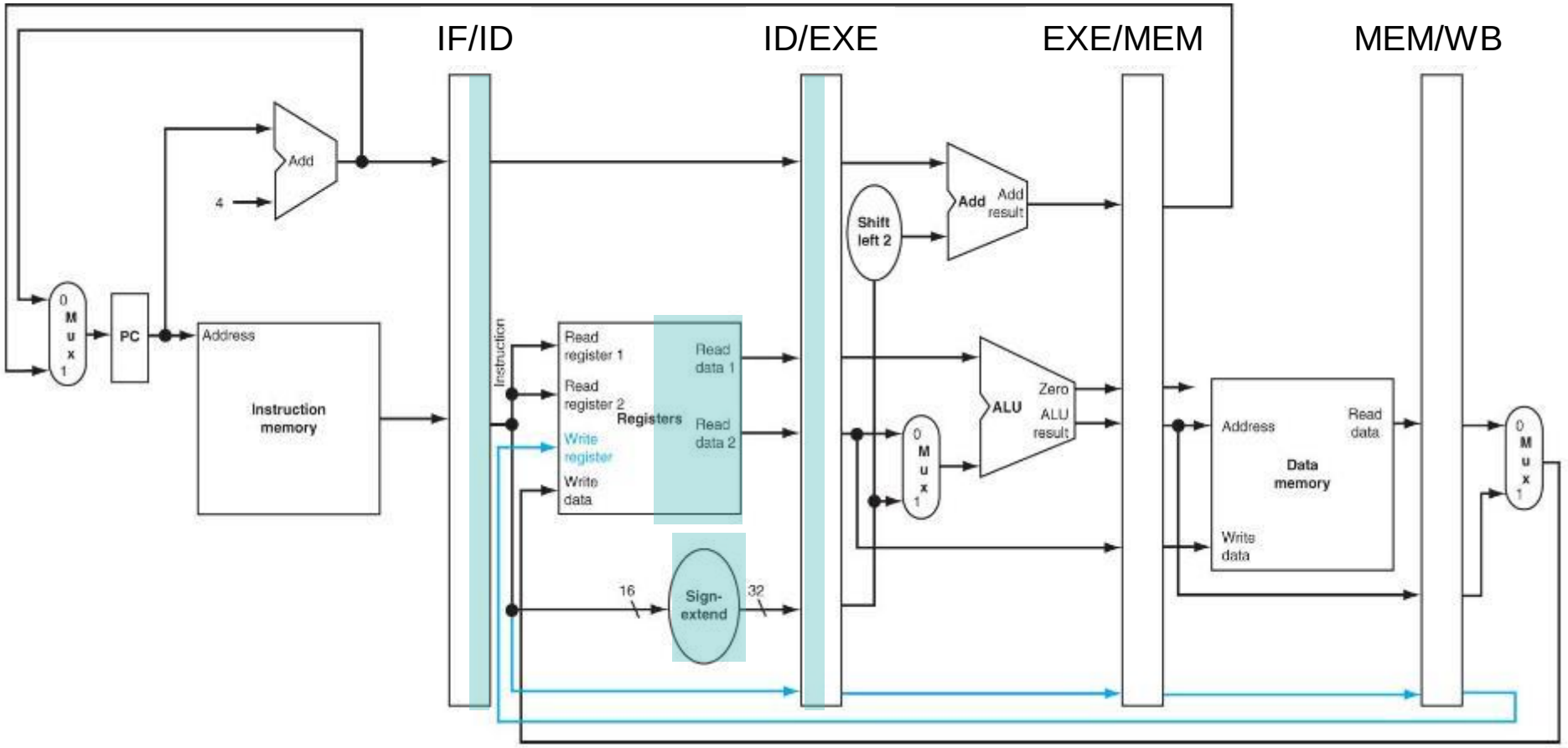
- **IF: Busca da instrução**



IF	ID	EXE	MEM	WB

Execução da instrução *sw*

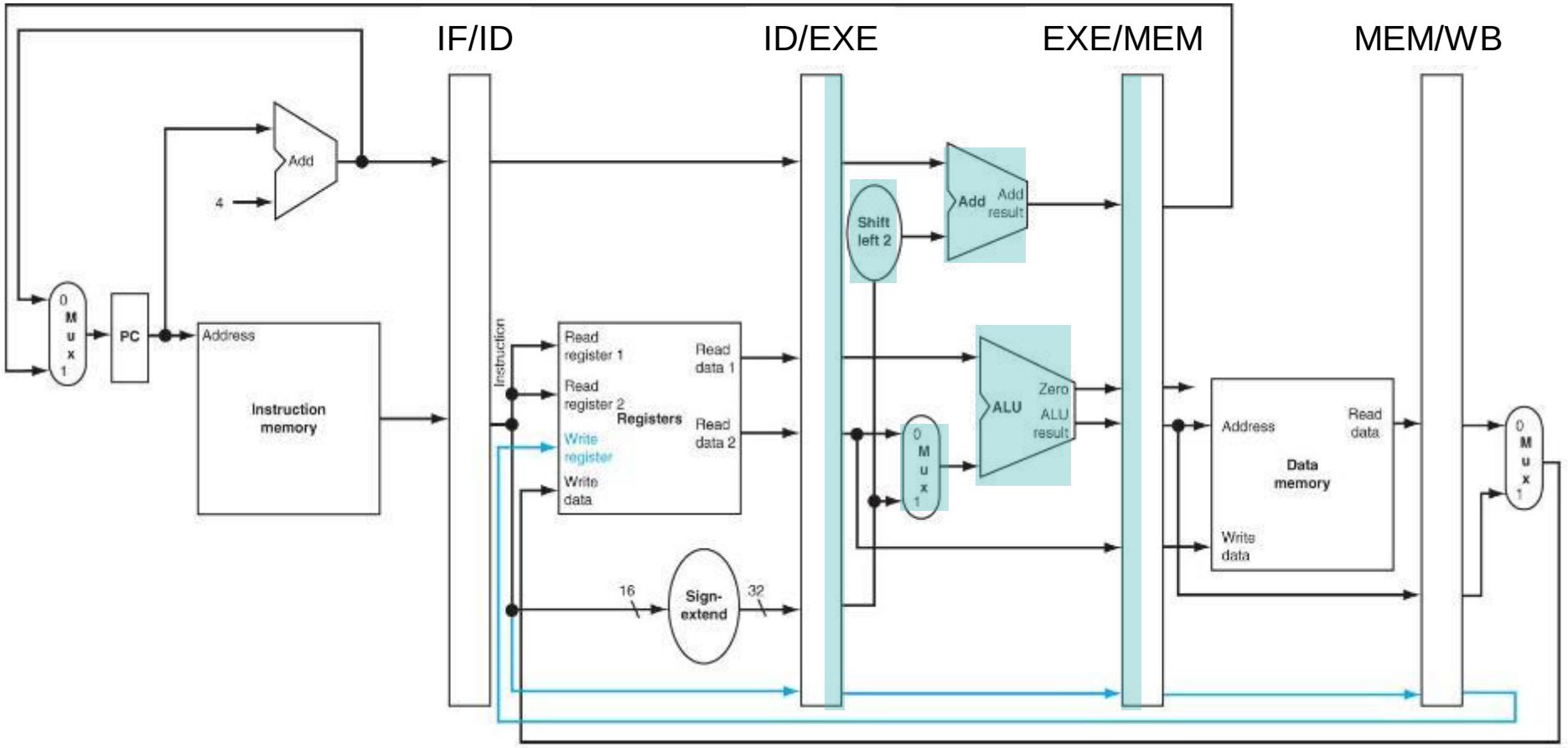
- **ID: Decodificação da instrução e busca dos operandos**



IF	ID	EXE	MEM	WB

Execução da instrução *sw*

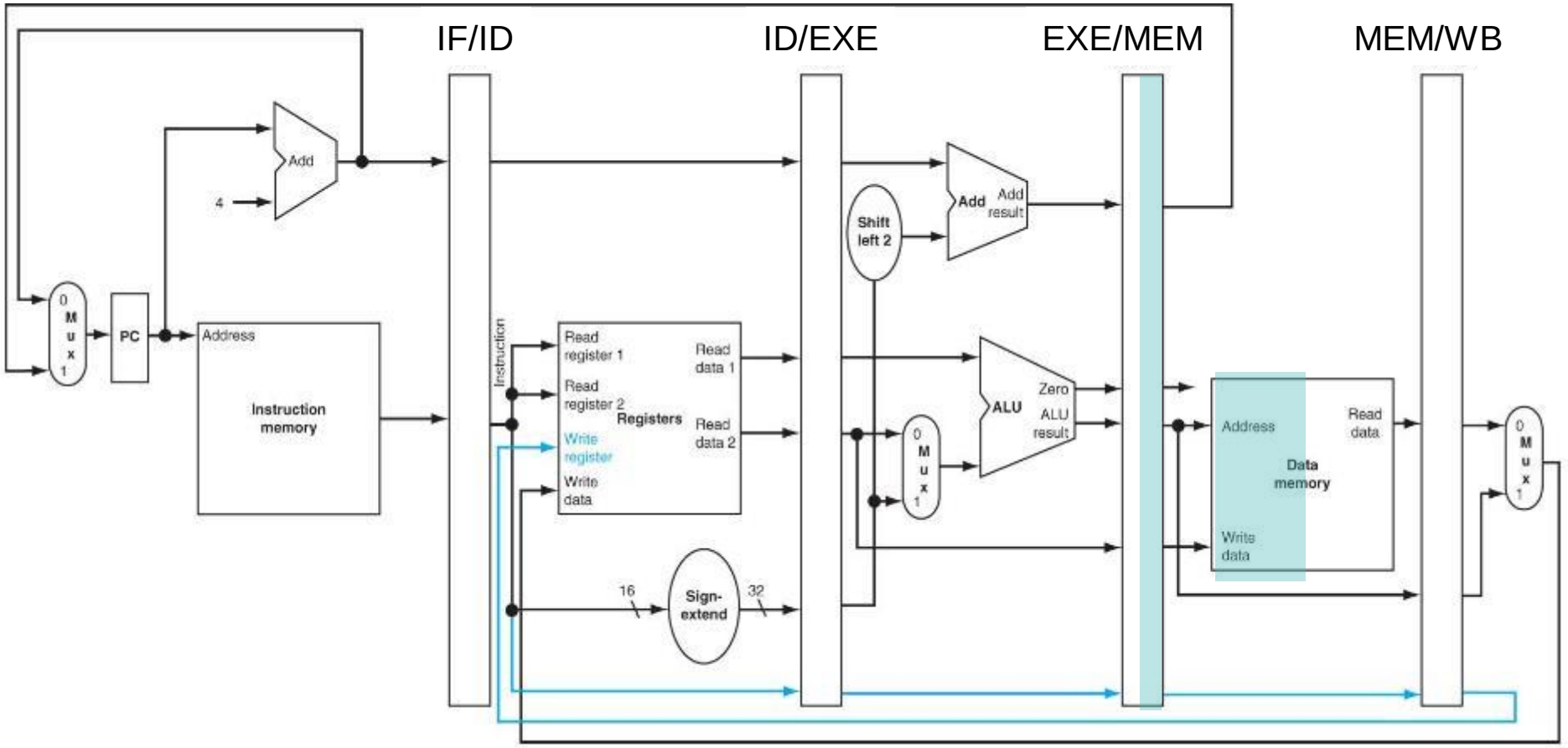
- **EXE: Cálculo do endereço efetivo**



IF	ID	EXE	MEM	WB

Execução da instrução *sw*

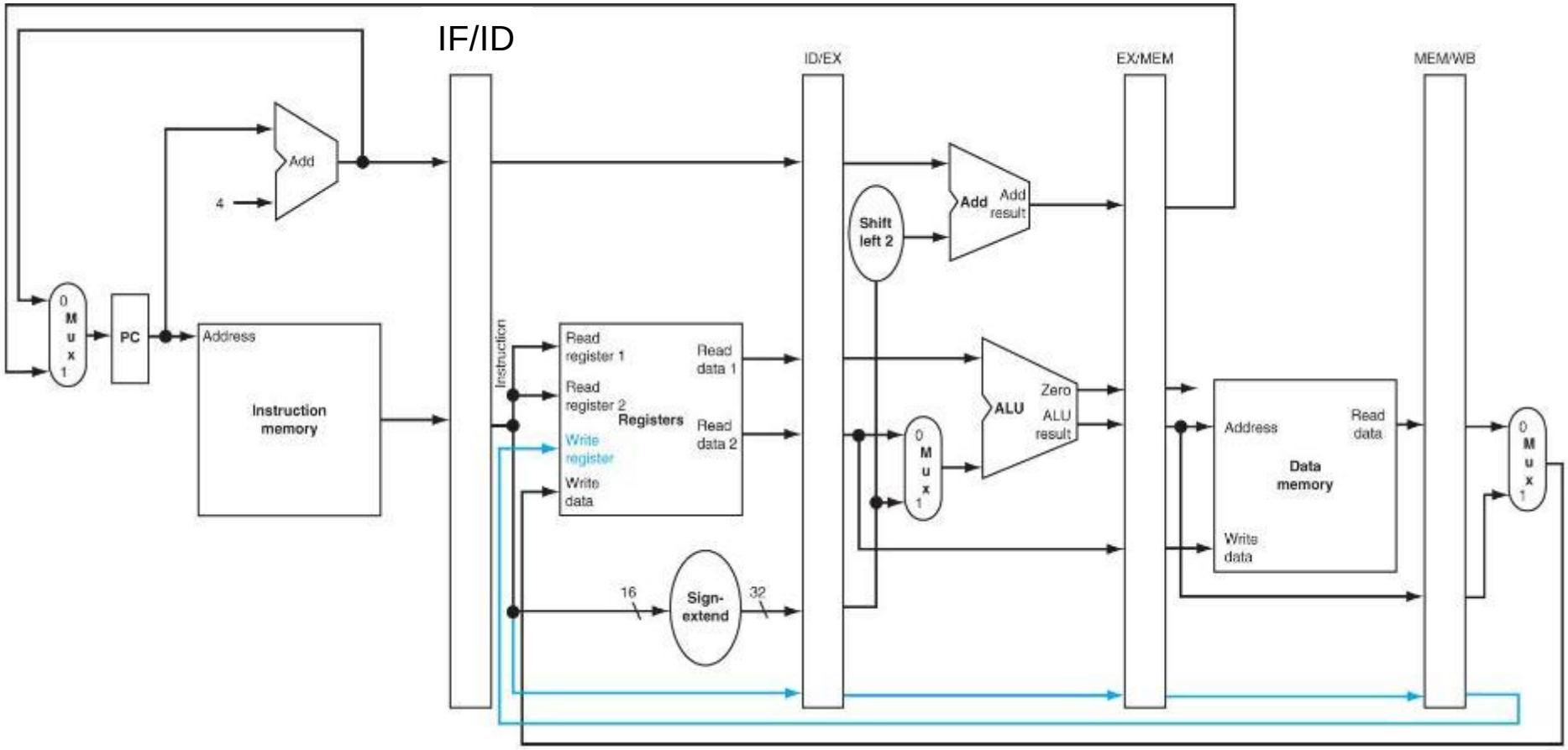
- **MEM: Acesso à memória**



IF	ID	EXE	MEM	WB

Execução da instrução *sw*

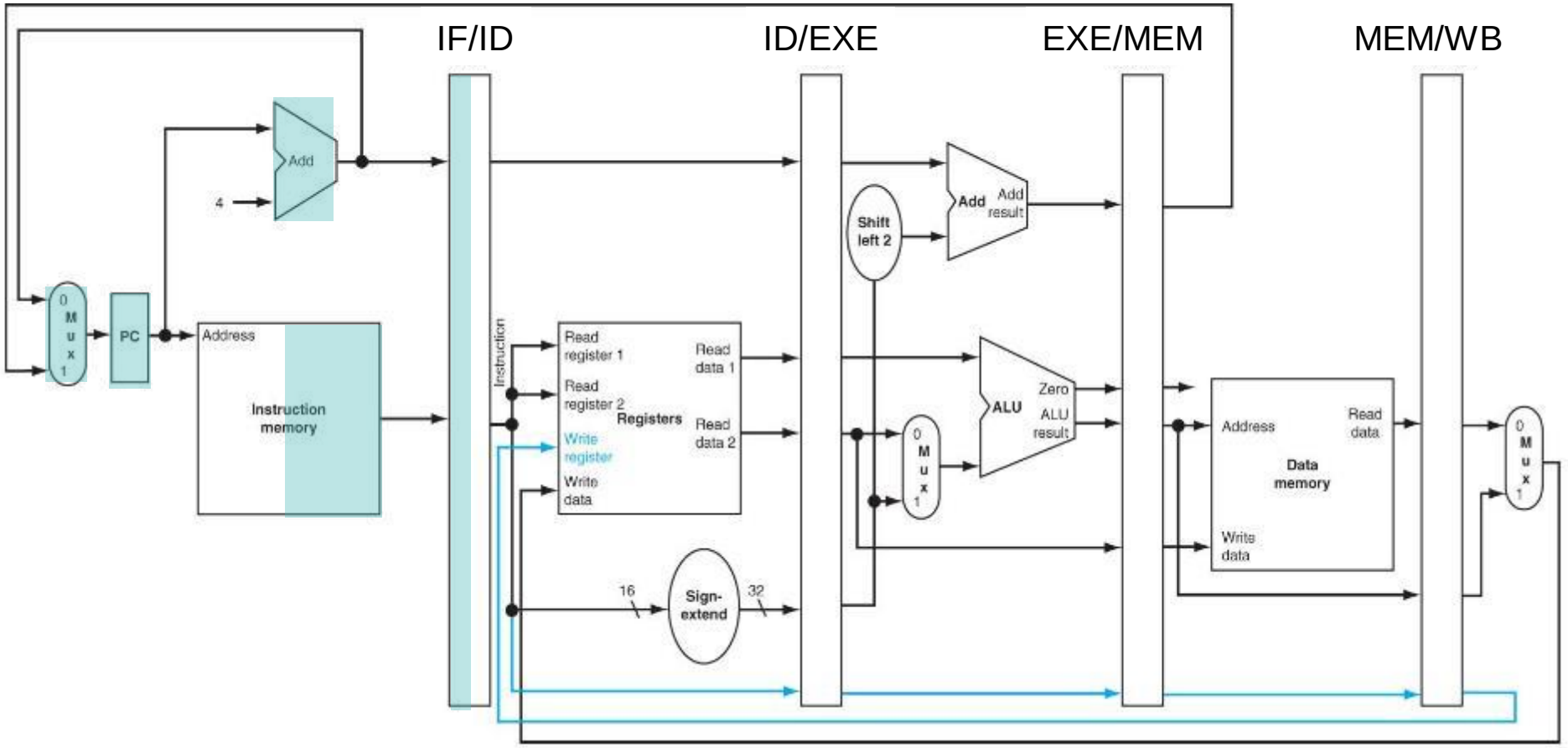
- **WB:** Não realiza nenhuma operação nesta etapa



IF	ID	EXE	MEM	WB

Execução da instrução *add*

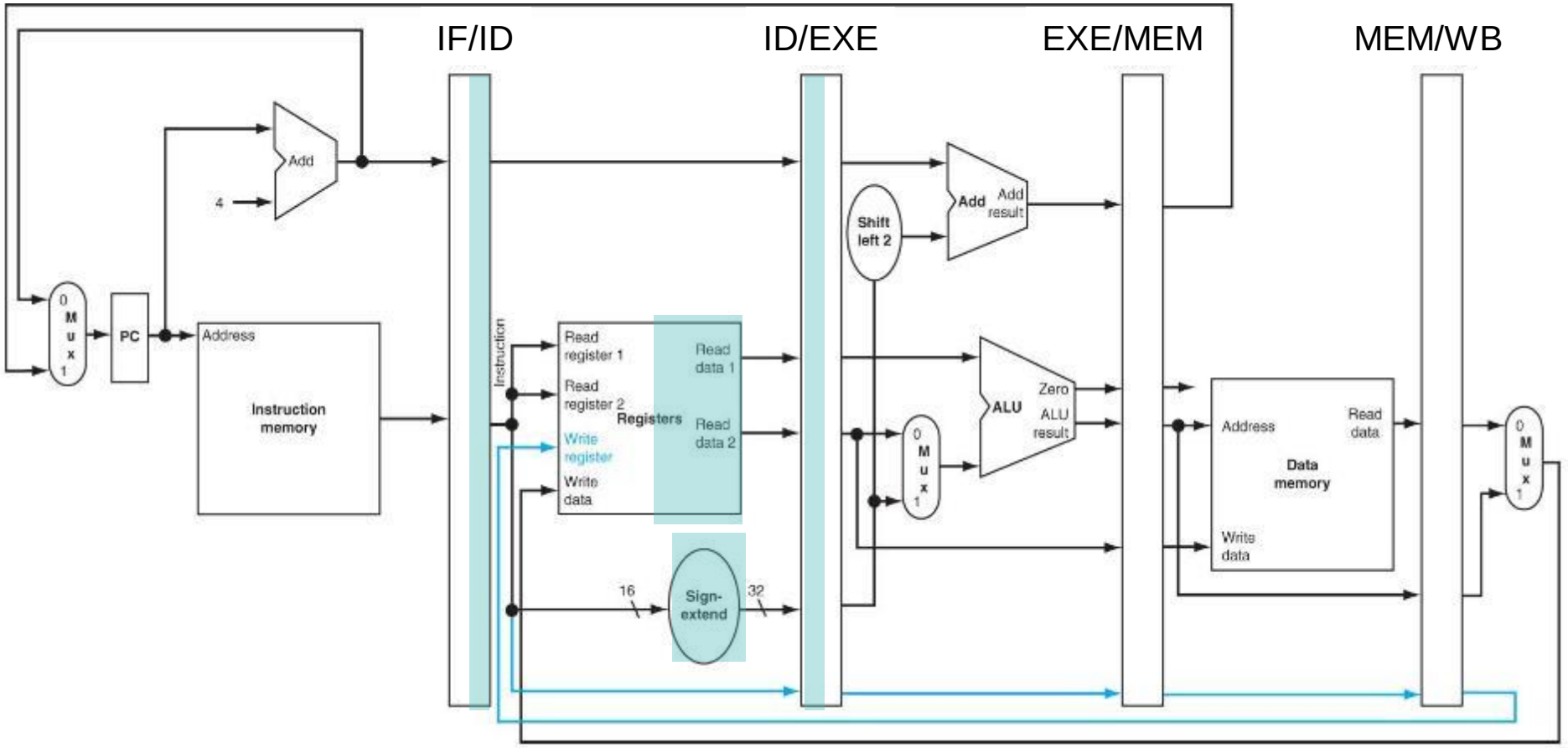
- **IF: Busca da instrução**



IF	ID	EXE	MEM	WB

Execução da instrução *add*

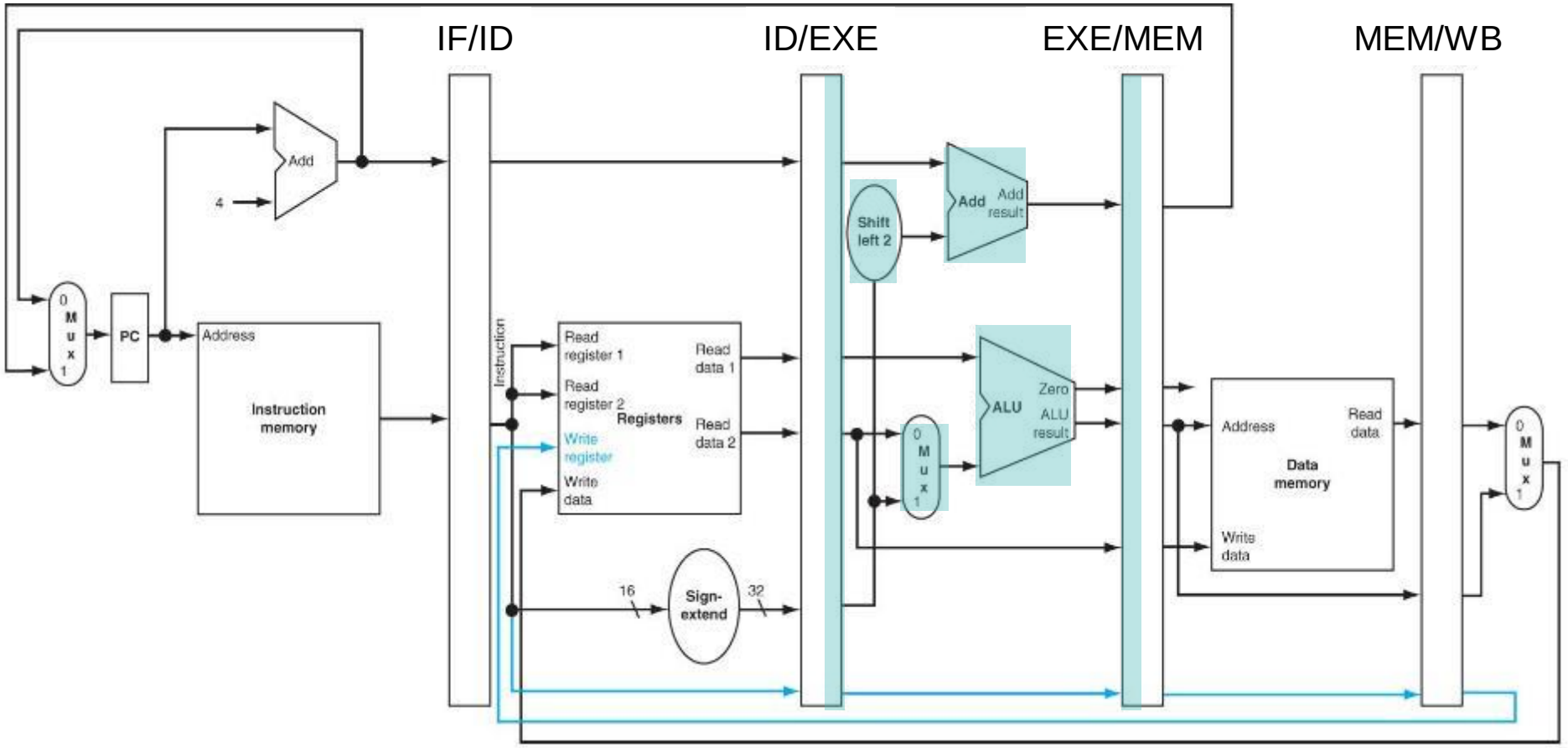
- **ID: Decodificação da instrução e busca dos operandos**



IF	ID	EXE	MEM	WB

Execução da instrução *add*

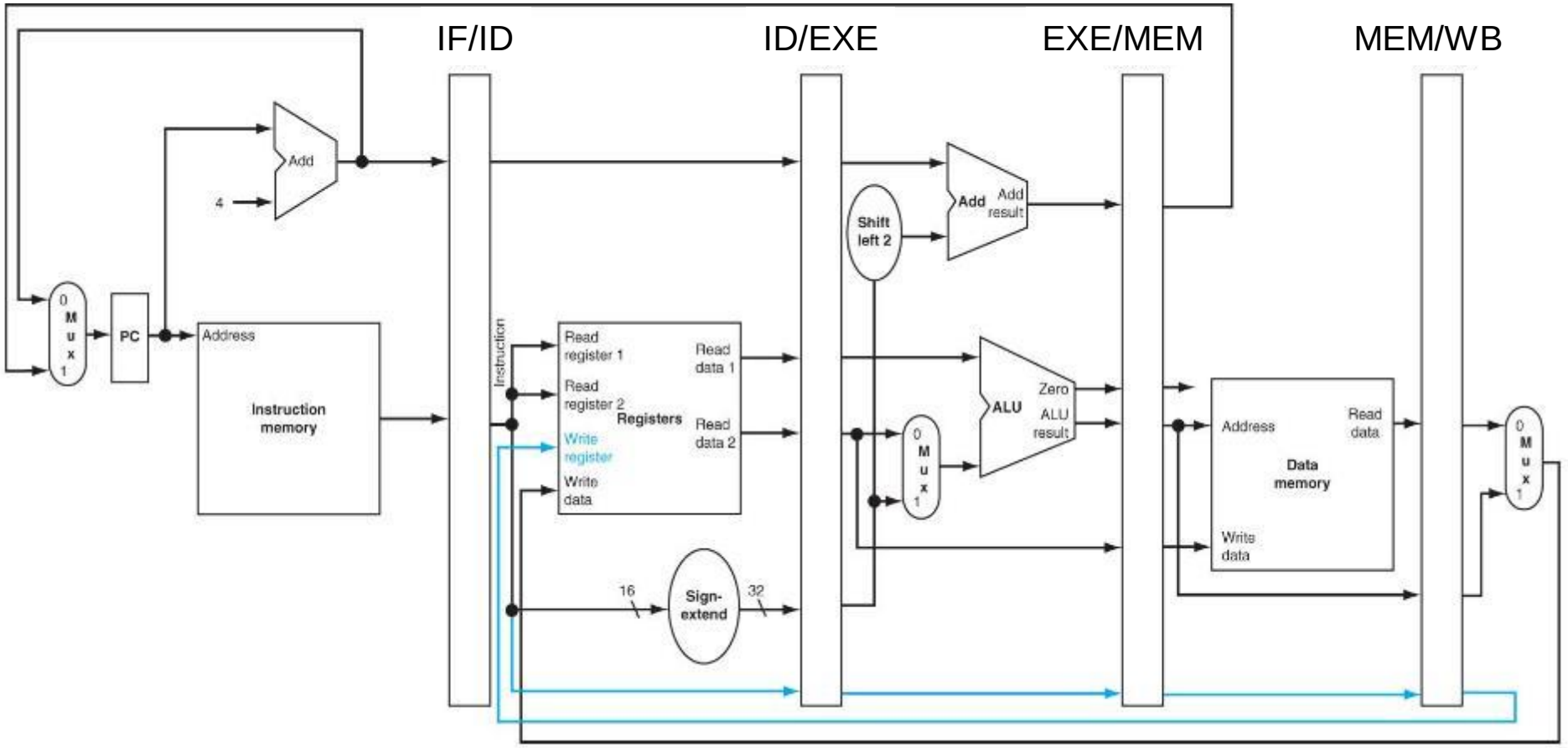
- **EXE: Execução da operação de soma**



IF	ID	EXE	MEM	WB

Execução da instrução *add*

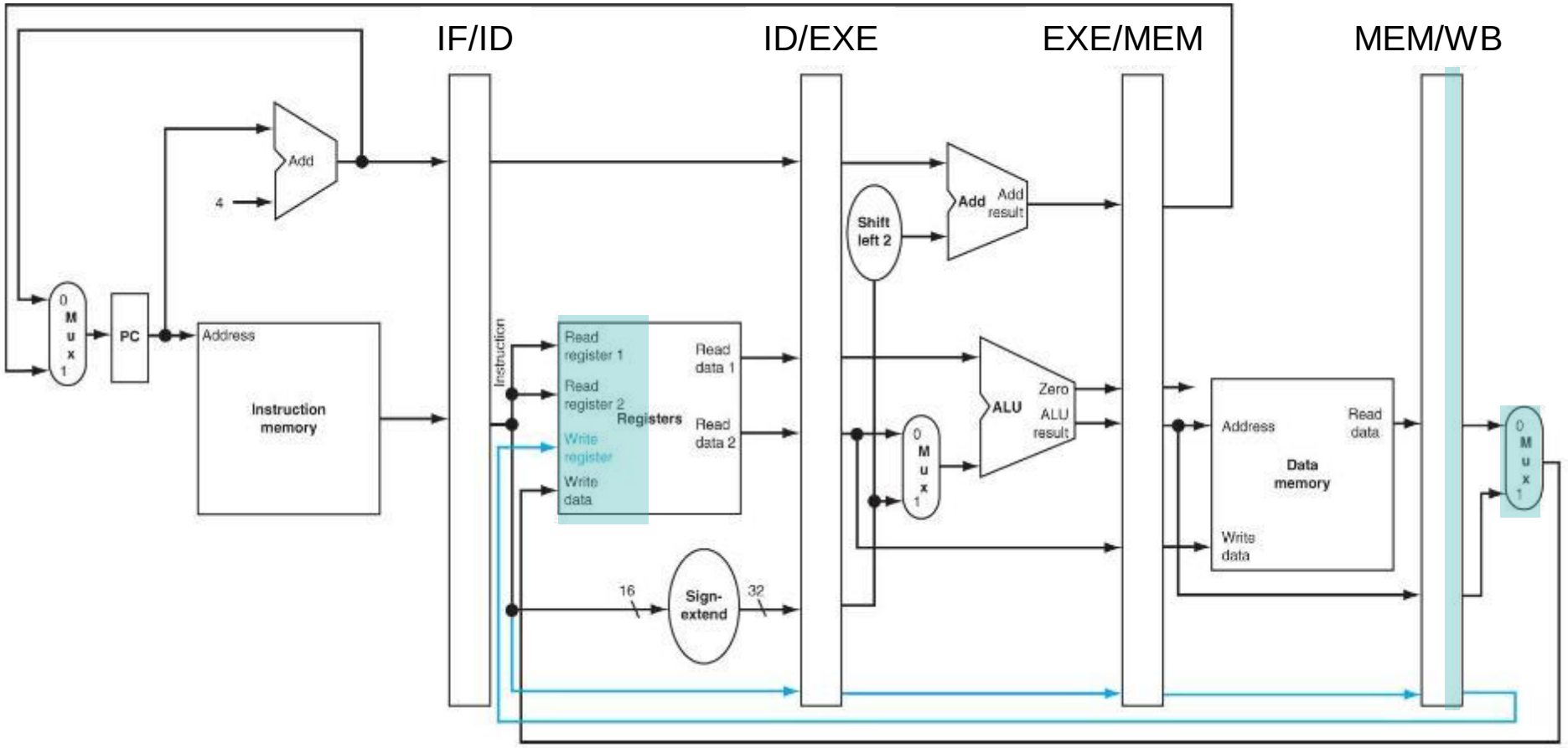
- **MEM: Não realiza nenhuma operação nesta etapa**



IF	ID	EXE	MEM	WB

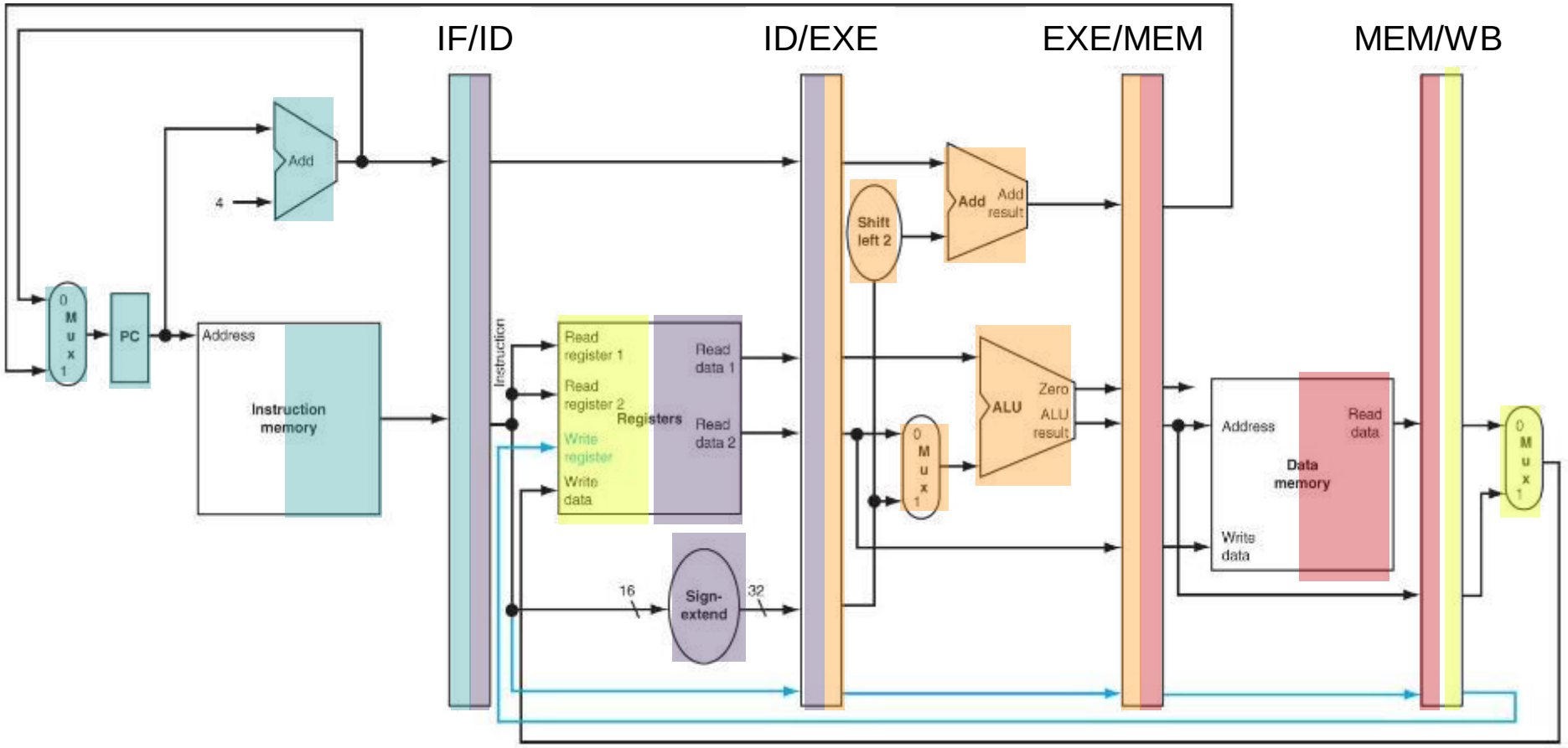
Execução da instrução *add*

- **WB: Escrita no registrador**



IF	ID	EXE	MEM	WB

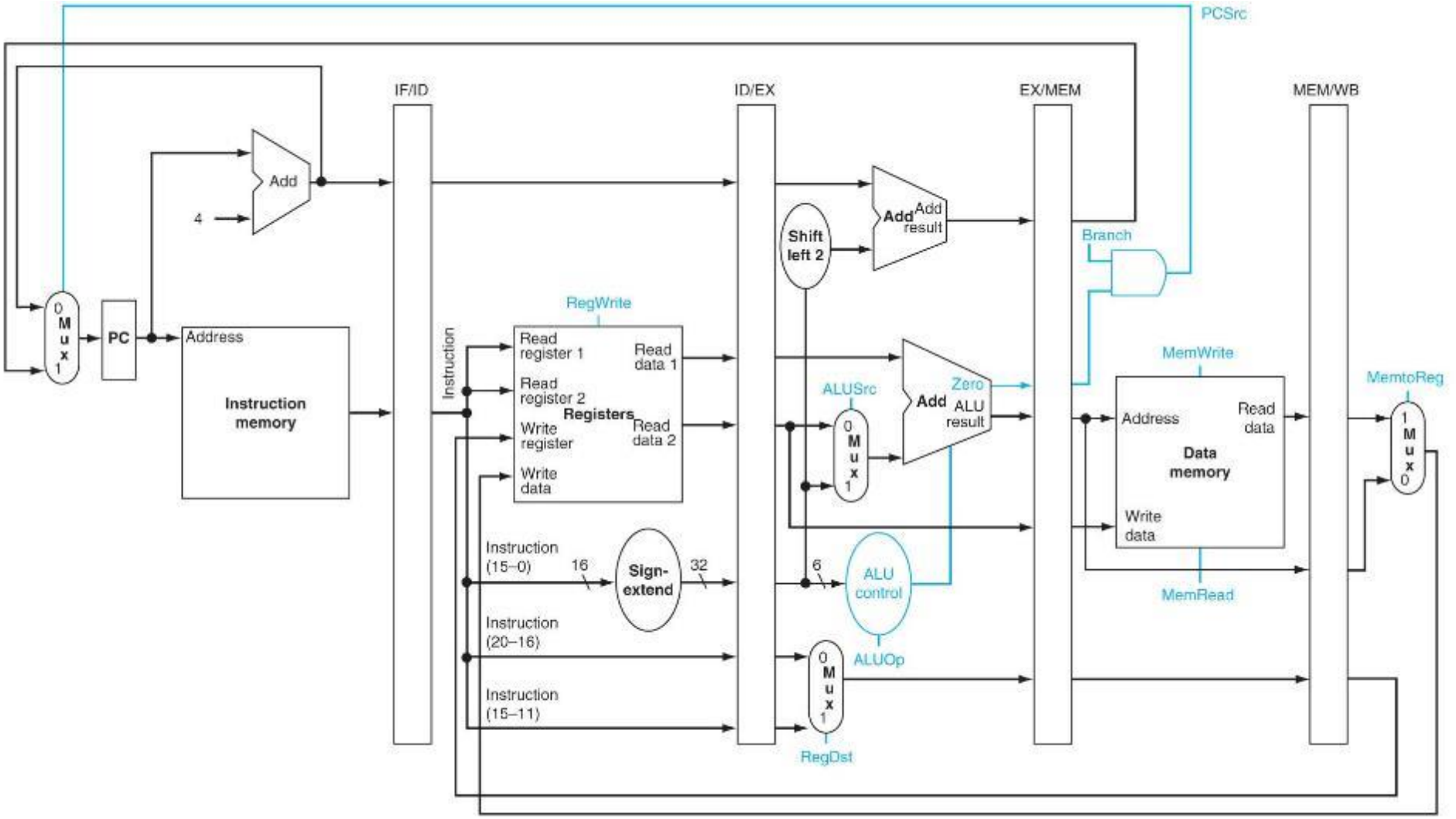
Execução de cinco instruções simultaneamente



IF	ID	EXE	MEM	WB

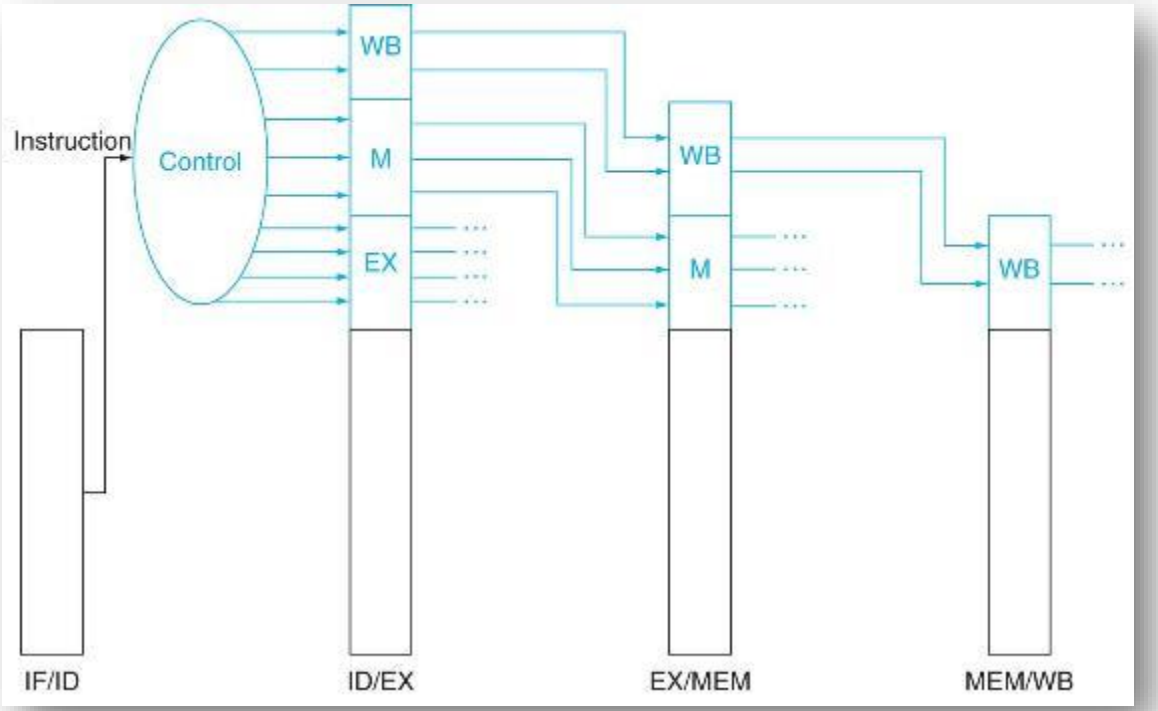
O controle

Sinais de controle



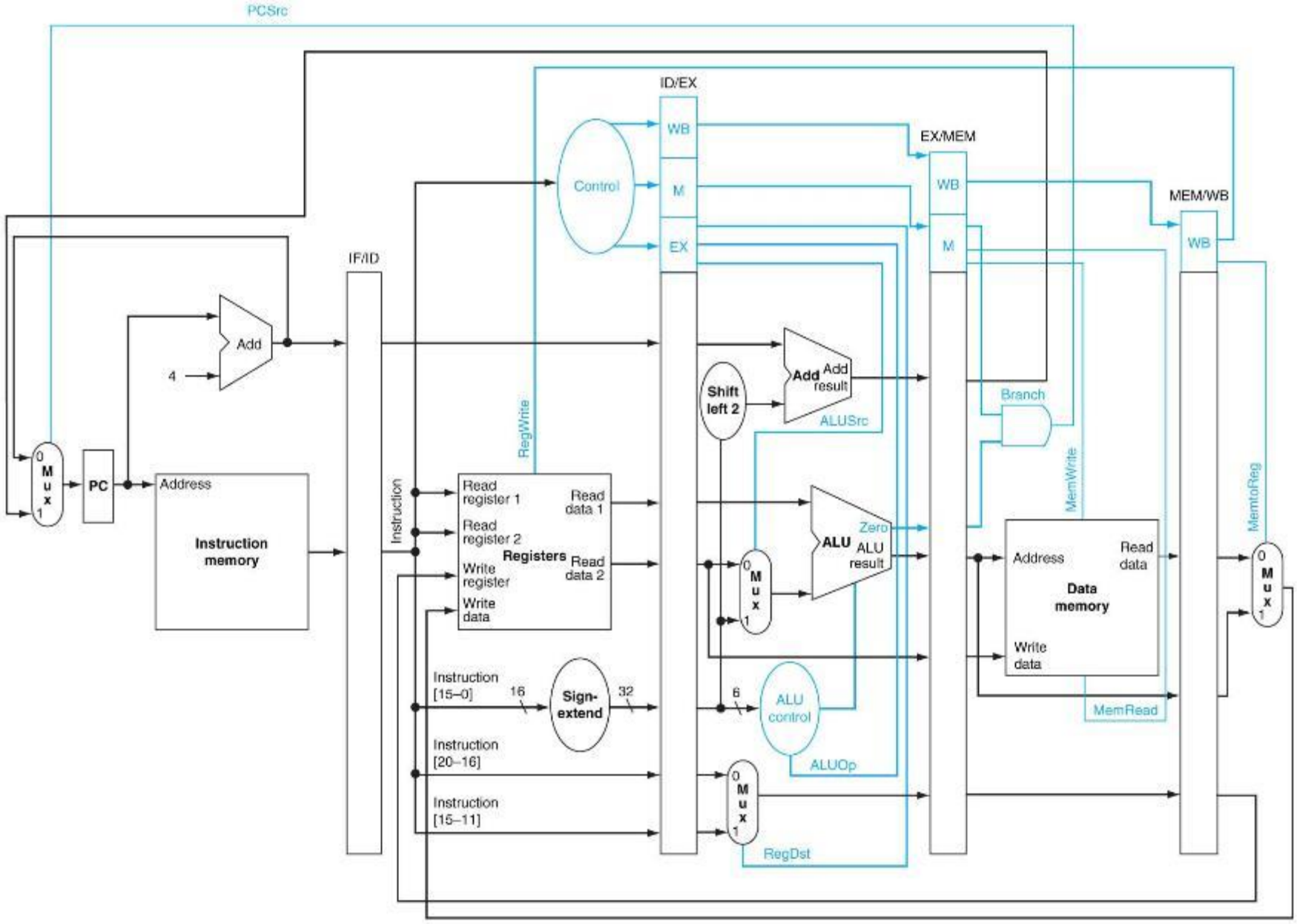
O controle

- **Os sinais de controle precisam ser propagados pelos registradores até o estágio em que são usados**
  - **Exemplo**: *RegWrite* e *MemToReg* têm que ser propagados até o estágio WB e armazenado nos registradores ID/EX, EX/MEM e MEM/WB



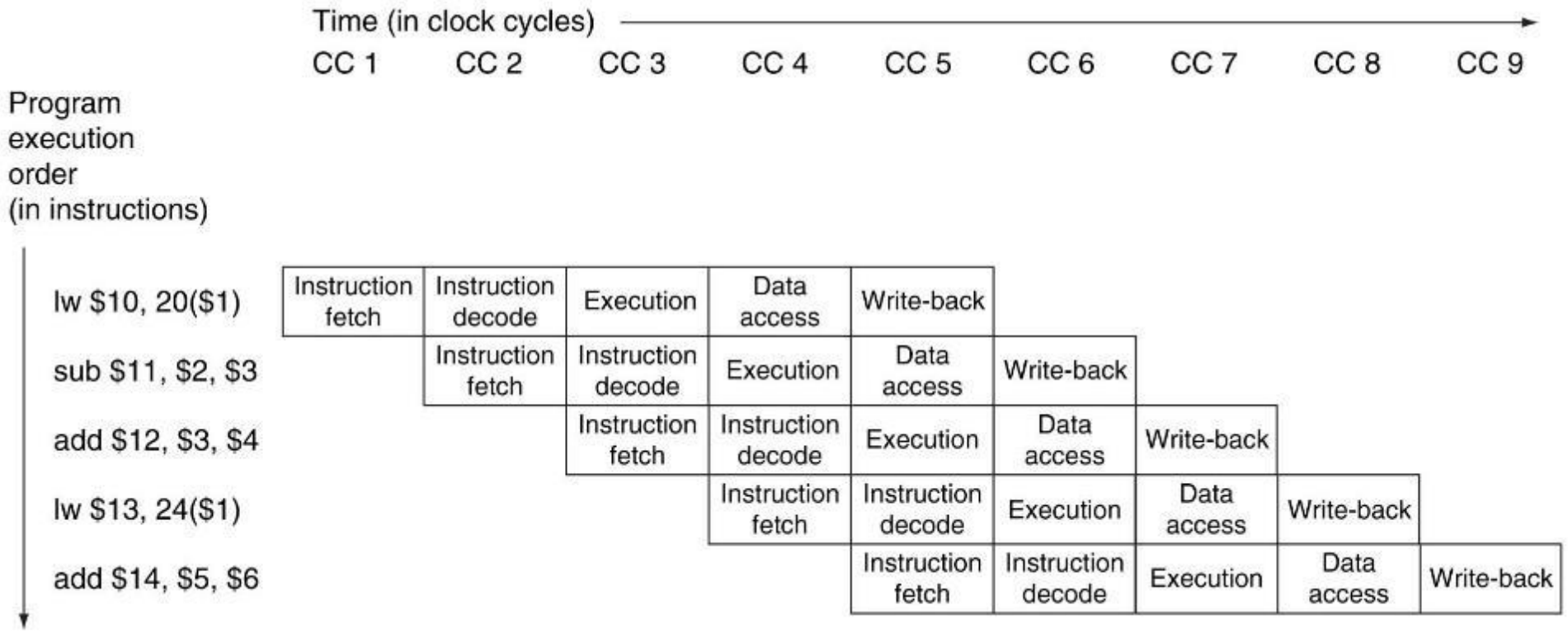
- **Custo com reg.**
  - IF/ID = 64 FFs
  - ID/EX = 133 + 9 FFs = 142 FFs
  - EX/MEM = 102 + 5 FFs = 107 FFs
  - MEM/WB= 69 + 2 FFs = 71 FFs

O controle



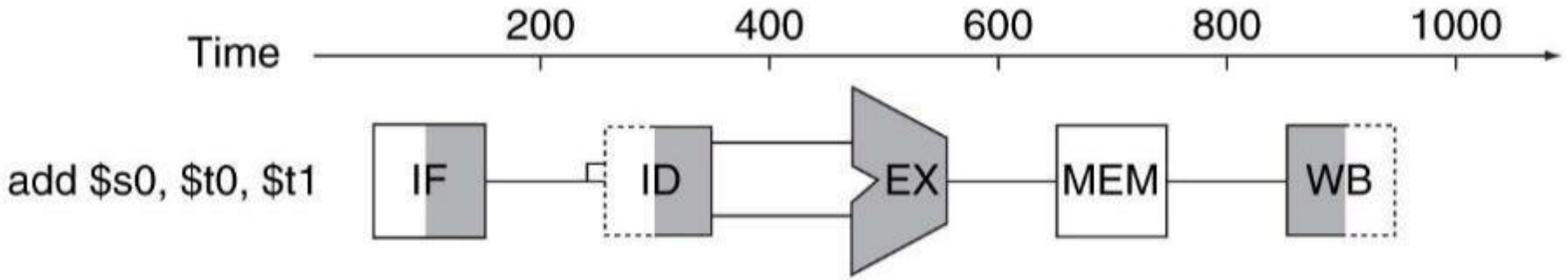
Representação do pipeline

Representação tradicional



Representação do pipeline

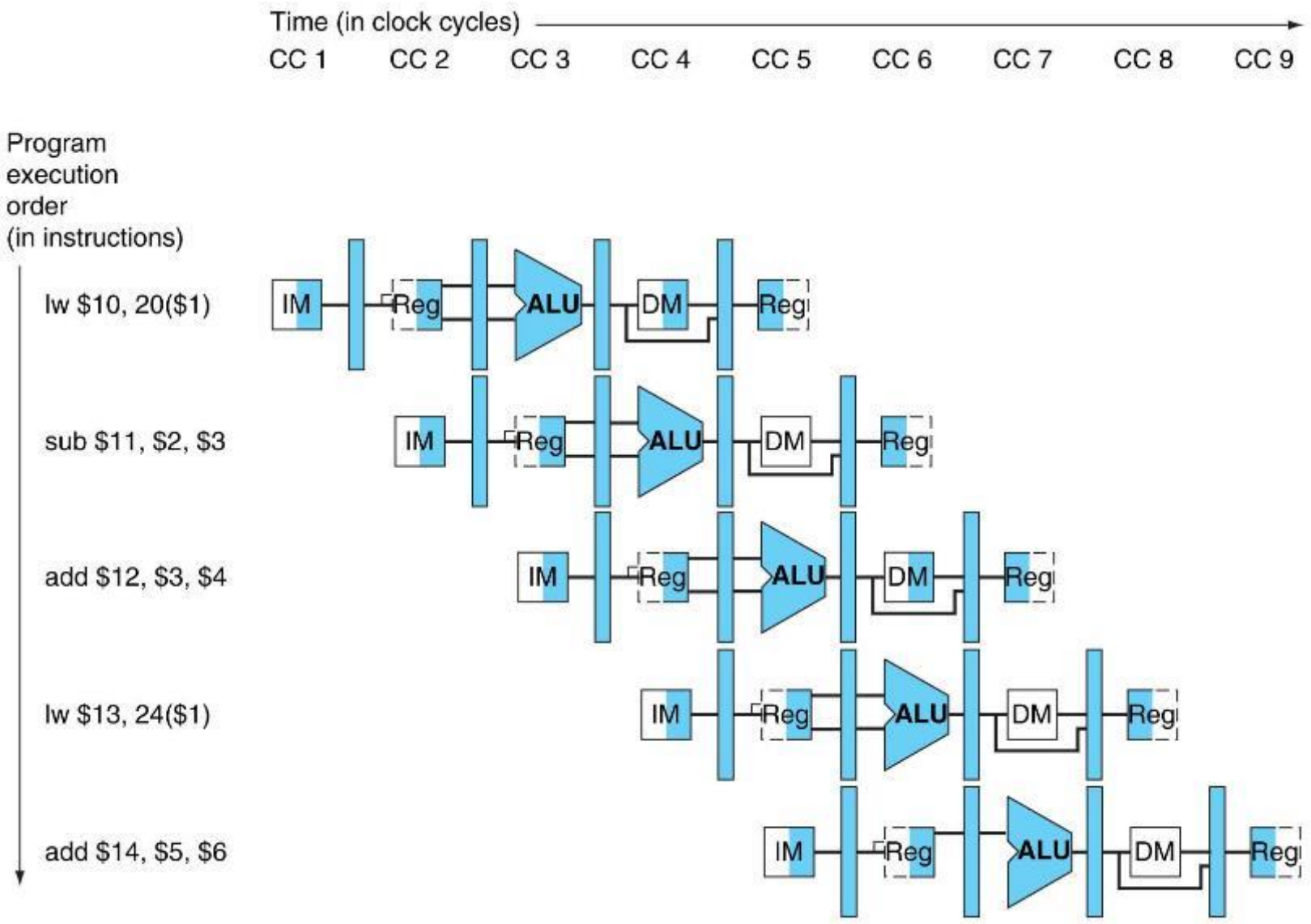
- **Representação alternativa**



- **Legenda**
  - **Leitura de memória/registrador**: preenchimento à direita
  - **Escrita em memória/registrador**: preenchimento à esquerda
  - **Processamento na ULA**: preenchimento completo
  - **Bloco não utilizado**: sem preenchimento

Representação do pipeline

- **Representação alternativa**

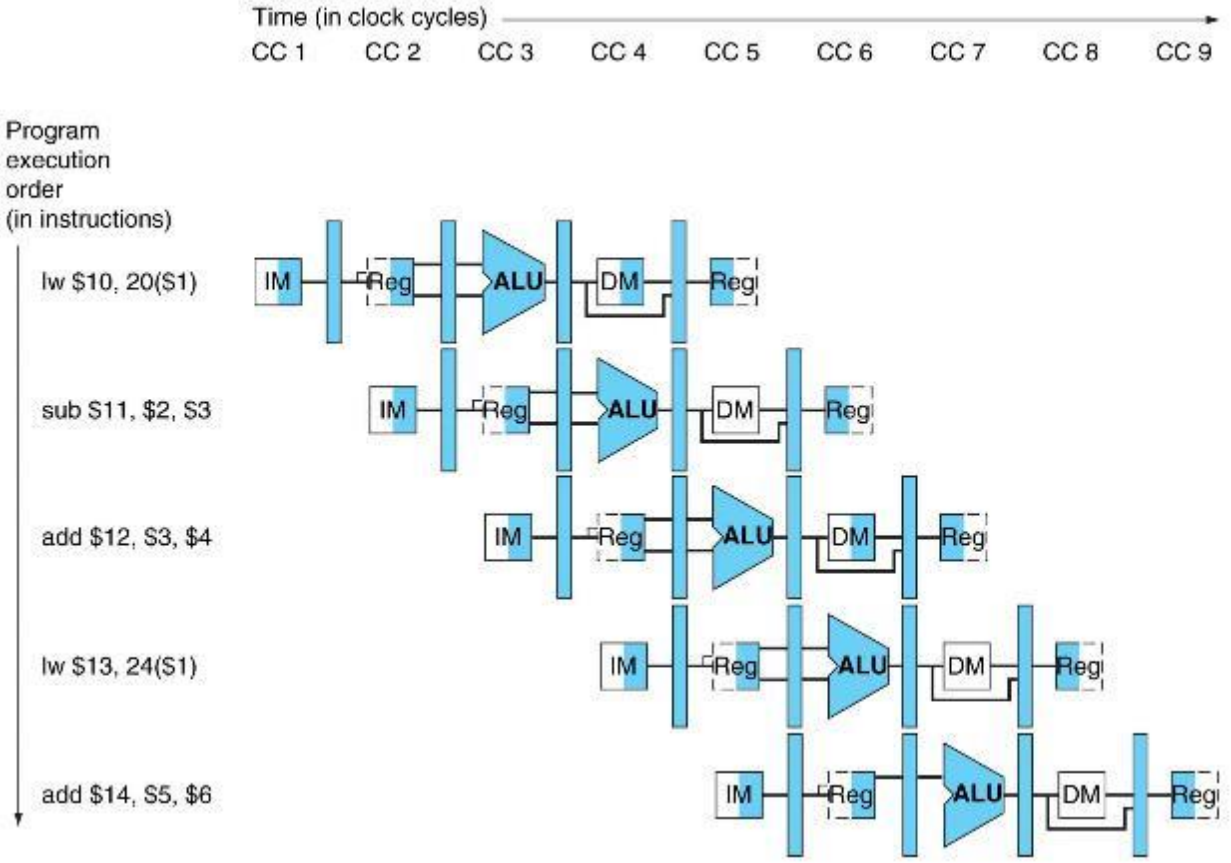


O pipeline real

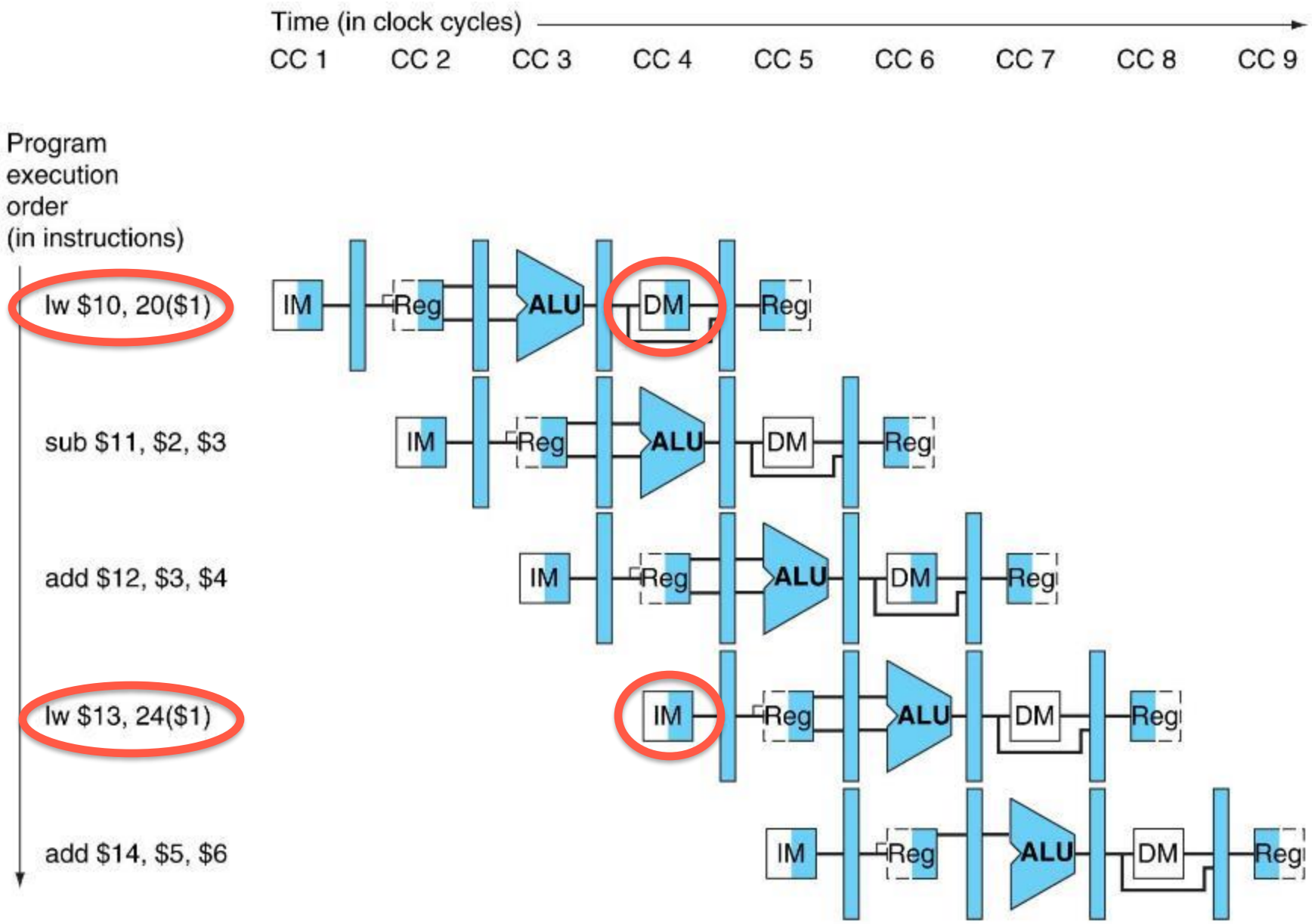
- **Conflitos (*hazards*) inviabilizam o pipeline ideal**
- **Tipos**
  - **Estrutural**: conflito no uso dos recursos pelas diferentes etapas do ciclo de instrução
  - **De dado**: dependência entre instruções sucessivas que acessam os mesmos registradores
  - **De controle**: alteração do fluxo do programa por instruções de desvio e chamada de funções

O pipeline real: conflito estrutural

- **Ocorre quando algum recurso não é suficientemente replicado**
- **Exemplo:** uma única memória para para dados e instruções

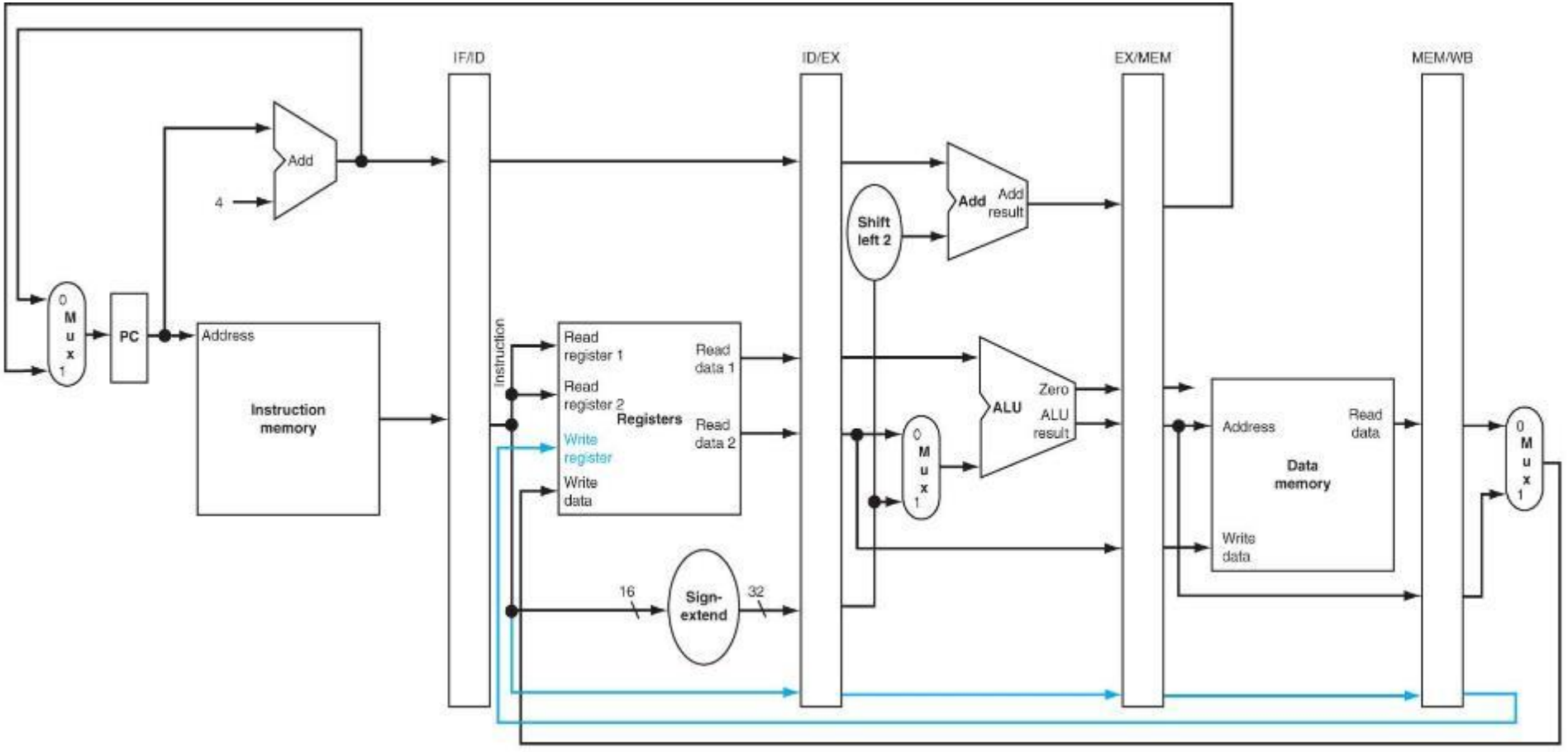


O pipeline real: conflito estrutural



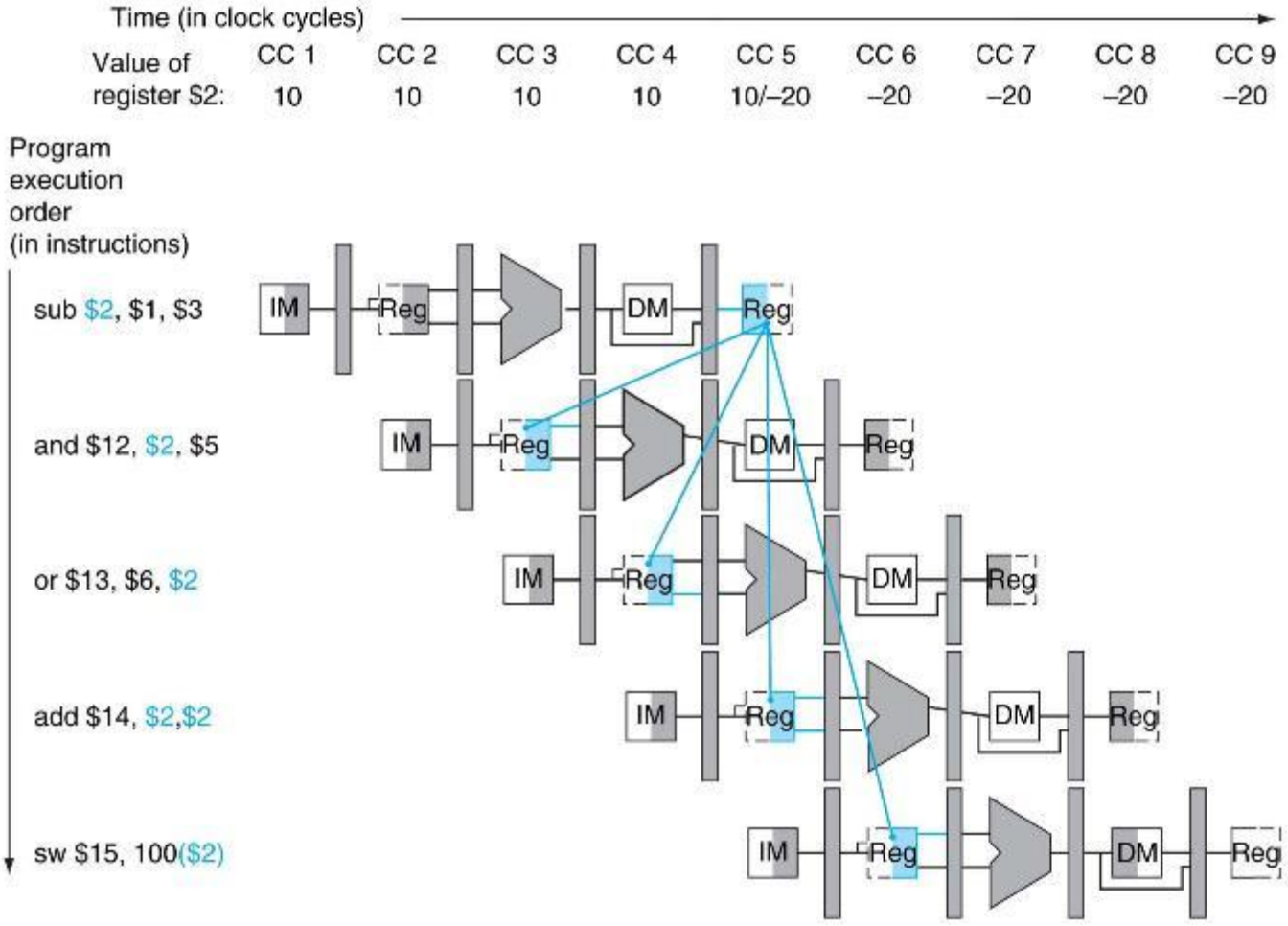
O pipeline real: conflito estrutural

- **Resolvido com o uso de arquitetura harvard**
  - Memórias separadas para instruções e dados no RISC-V monociclo
  - Possibilita a busca de instrução e o acesso a uma variável por outra instrução no mesmo ciclo de relógio

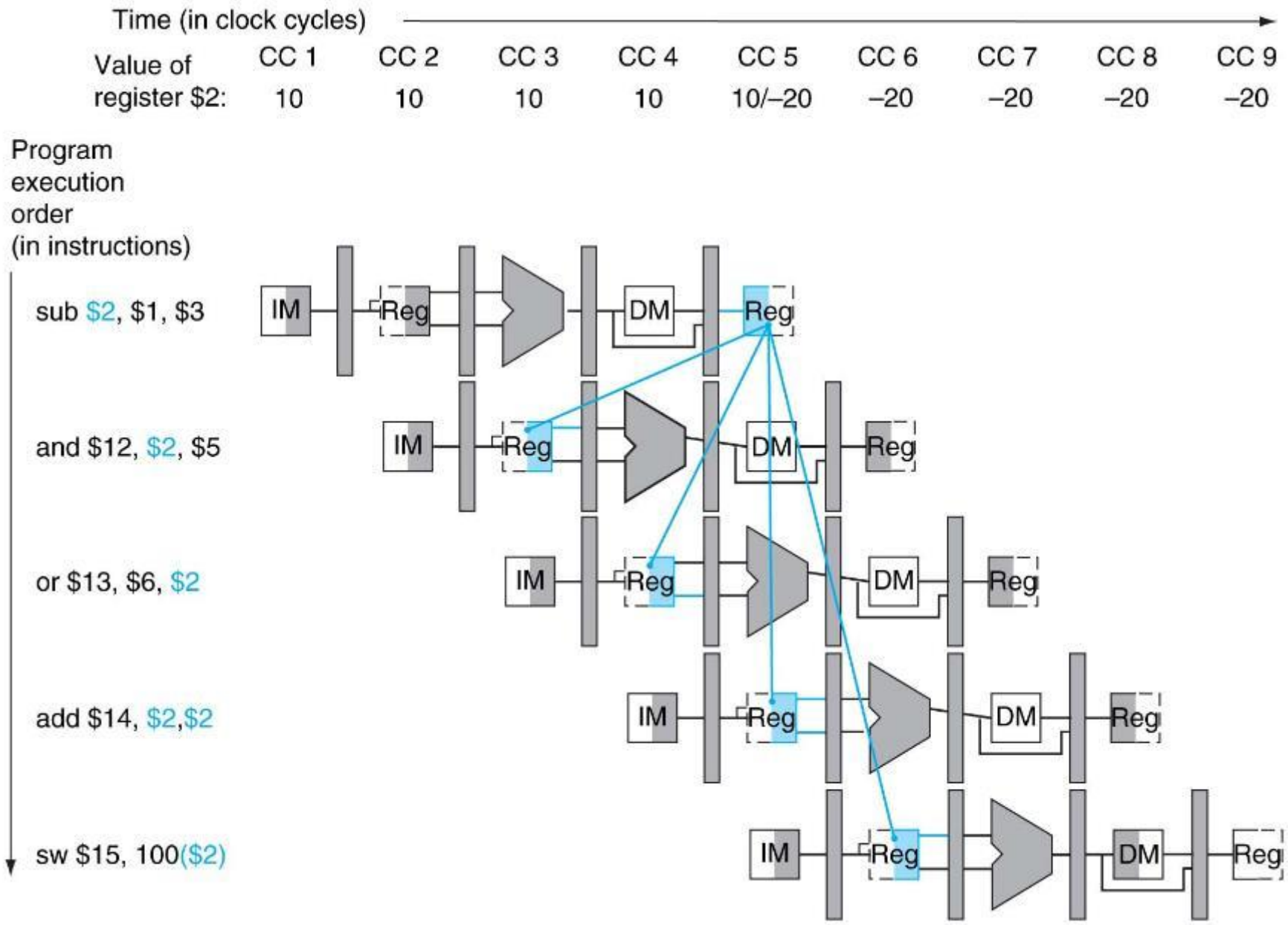


O pipeline real: conflito de dado

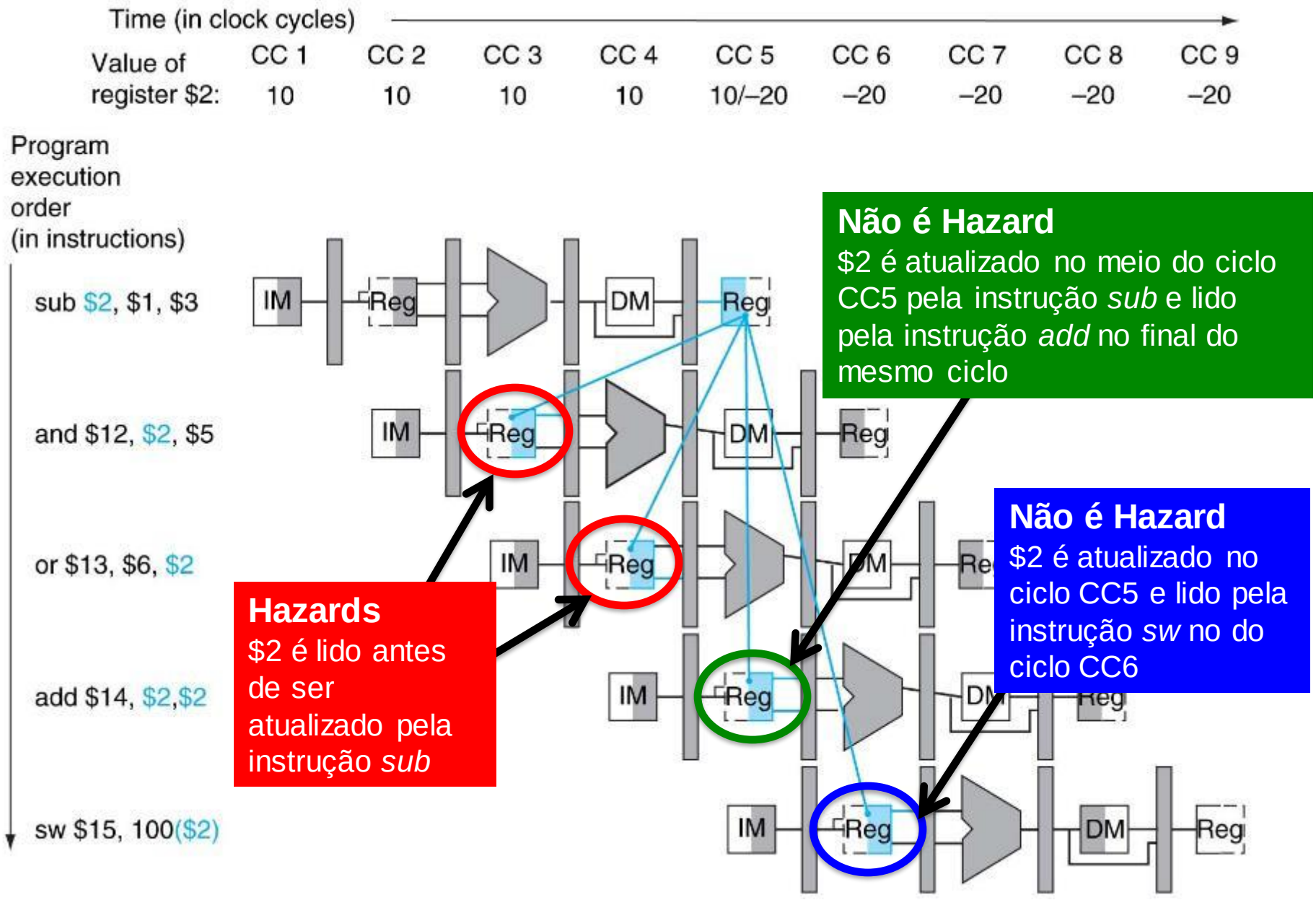
- Ocorre quando instruções lêem um registrador a ser atualizado por uma instrução anterior antes da atualização ser realizada
- Exemplo



O pipeline real: conflito de dado



O pipeline real: conflito de dado



O pipeline real: conflito de dado

- **Soluções**
  1. **Inserção de NOPs**
  2. **Reordenamento de instruções**
  3. **Método do adiantamento (*forwarding*)**
  4. **Inserção de bolhas (*stalls*)**

O pipeline real: conflito de dado

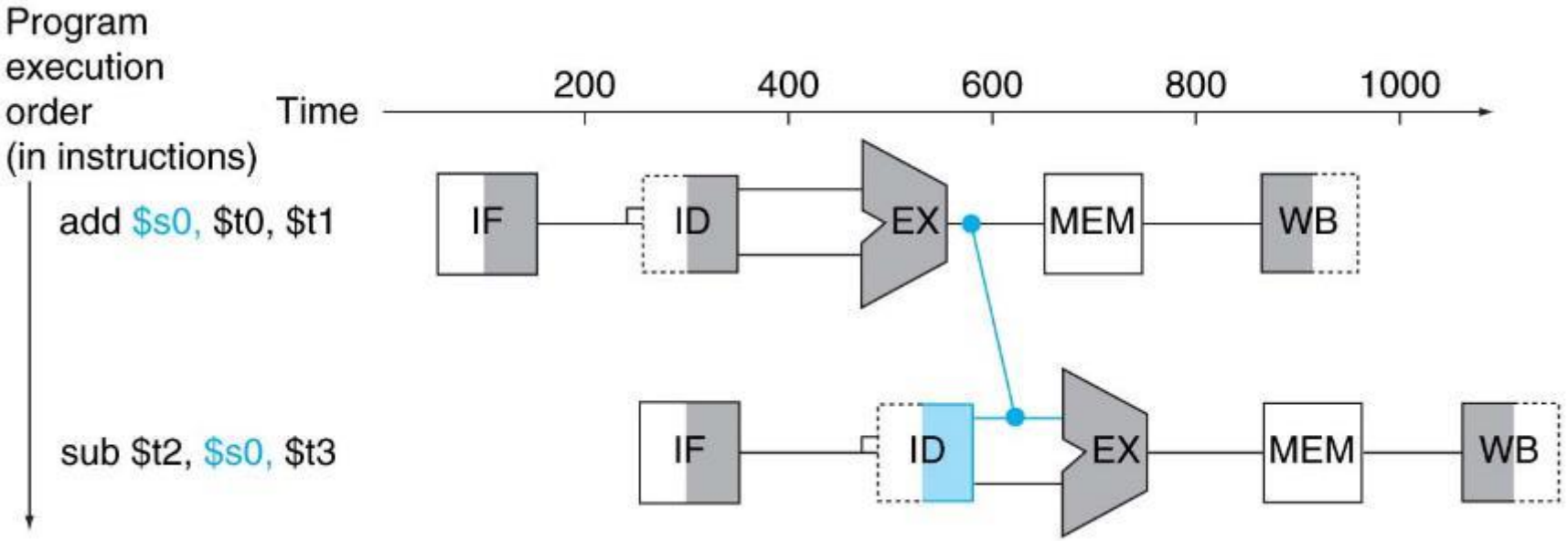
- **Inserção de NOPs**
  - *nop* é uma pseudo-instrução que não modifica o estado de nenhum registrador do processador (salvo o PC). Ela pode ser implementada por uma instrução de aritmética ou de lógica que tenha o registrador $zero como destino
  - Ao inserir *nop*s atrasa-se a busca das instruções que geram *hazard* de dados
  - Resulta em degradação de desempenho

O pipeline real: conflito de dado

- **Reordenamento de instruções**
  - Aos invés de *nops*, pode ser antecipada a execução de instruções posteriores que não tenham dependência com nenhuma instrução anterior a elas e que não modifiquem nenhum registrador lido por tais instruções
  - Exige um compilador inteligente

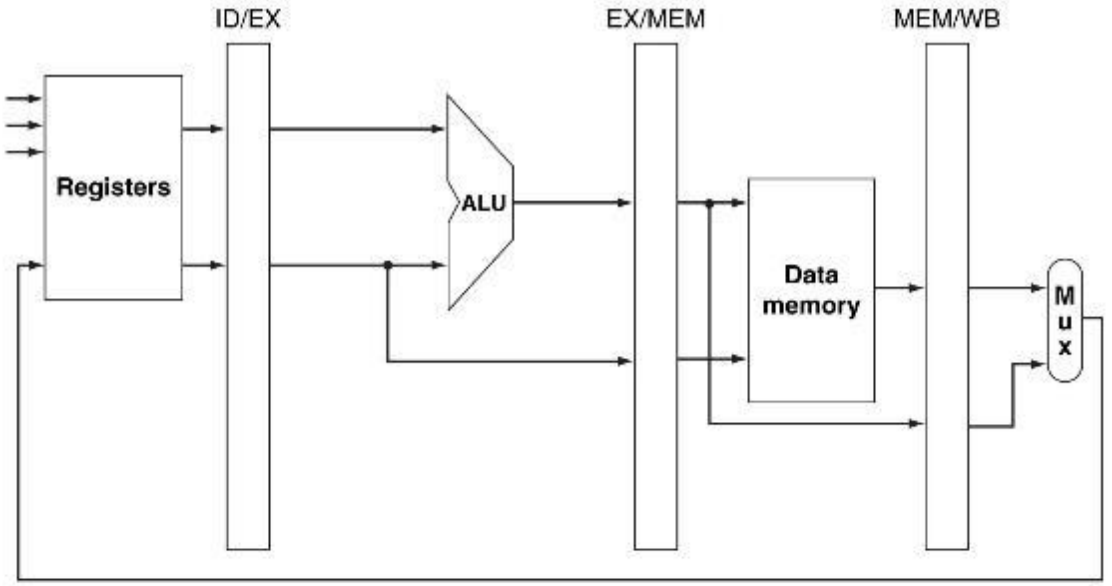
O pipeline real: conflito de dado

- **Método de adiantamento (*forwarding*) ou curto-circuito**
  - Solução de hardware baseada em caminhos internos que retro-propagam os resultados de um estágio para estágio(s) anterior(es) do pipeline antes da atualização do destino

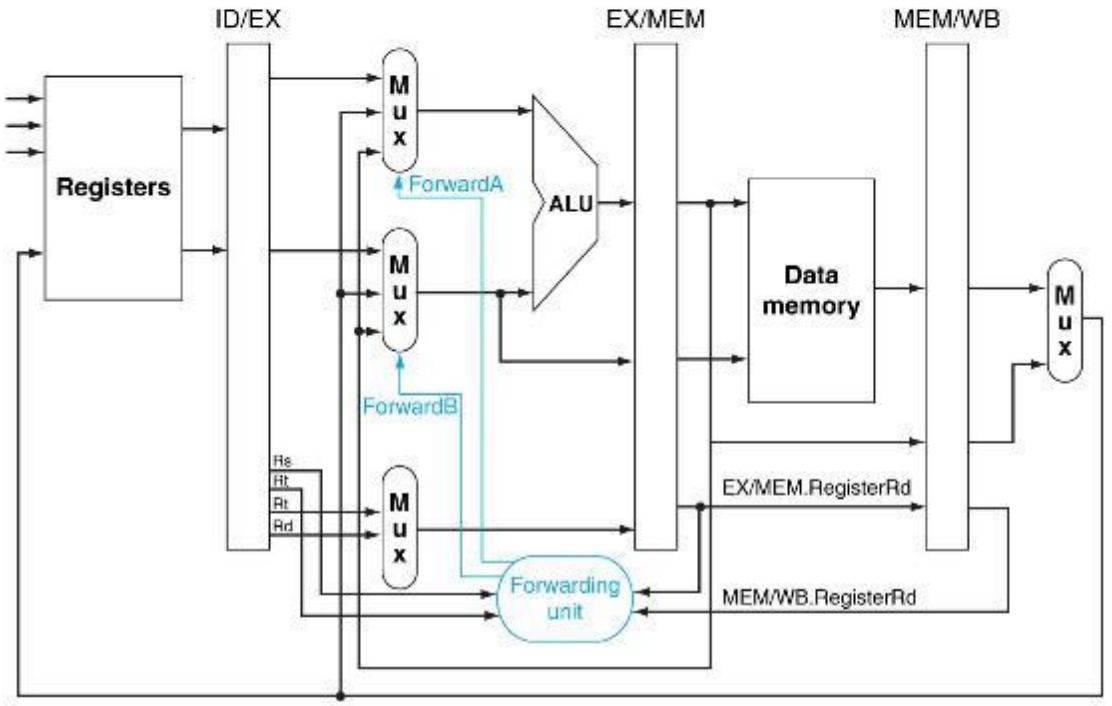


O pipeline real: confli

- **A unidade de adiantamento**
  - Circuito que compara os endereços dos registradores destino (*rd*) das instruções que estão no estágios MEM e WB (se forem escrever em registrador) com os endereços dos registradores fonte (*rs0* e *rs1*) da instrução que está no estágio EX
  - Comanda dois multiplexadores que adianta o resultado dos estágios MEM ou WB para o estágio EX caso seja necessário



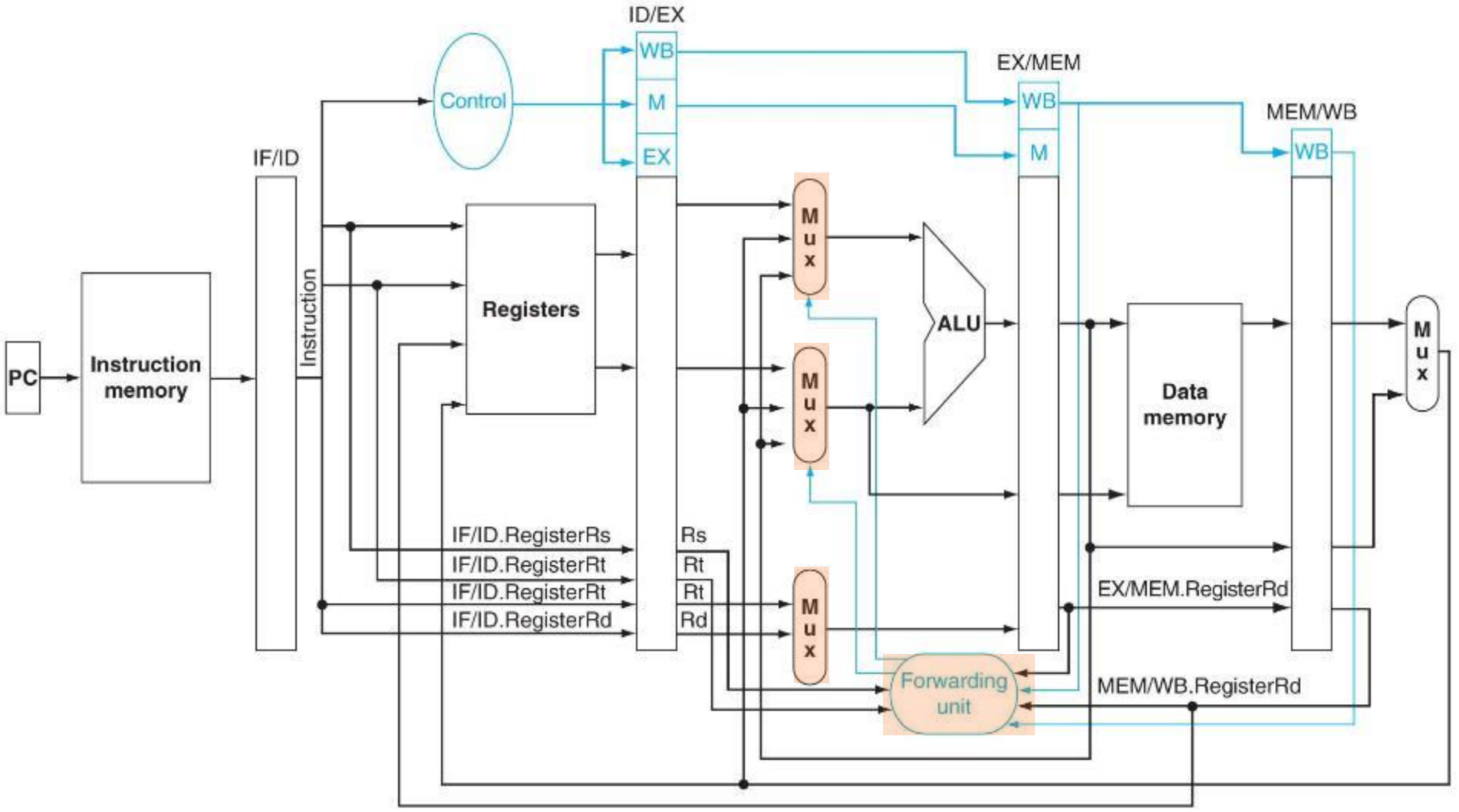
a. No forwarding



b. With forwarding

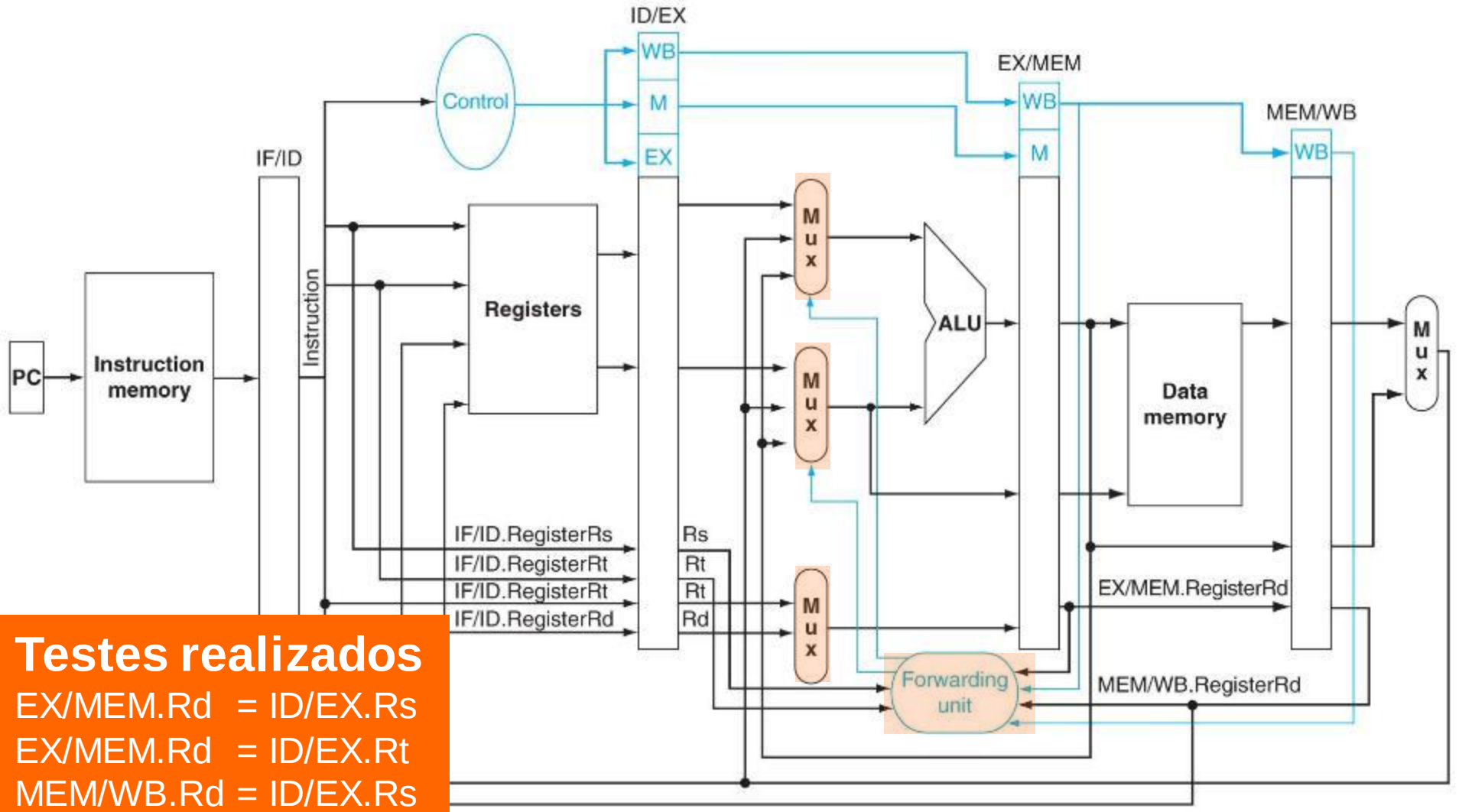
O pipeline real: conflito de dado

- A unidade de adiantamento



O pipeline real: conflito de dado

- A unidade de adiantamento

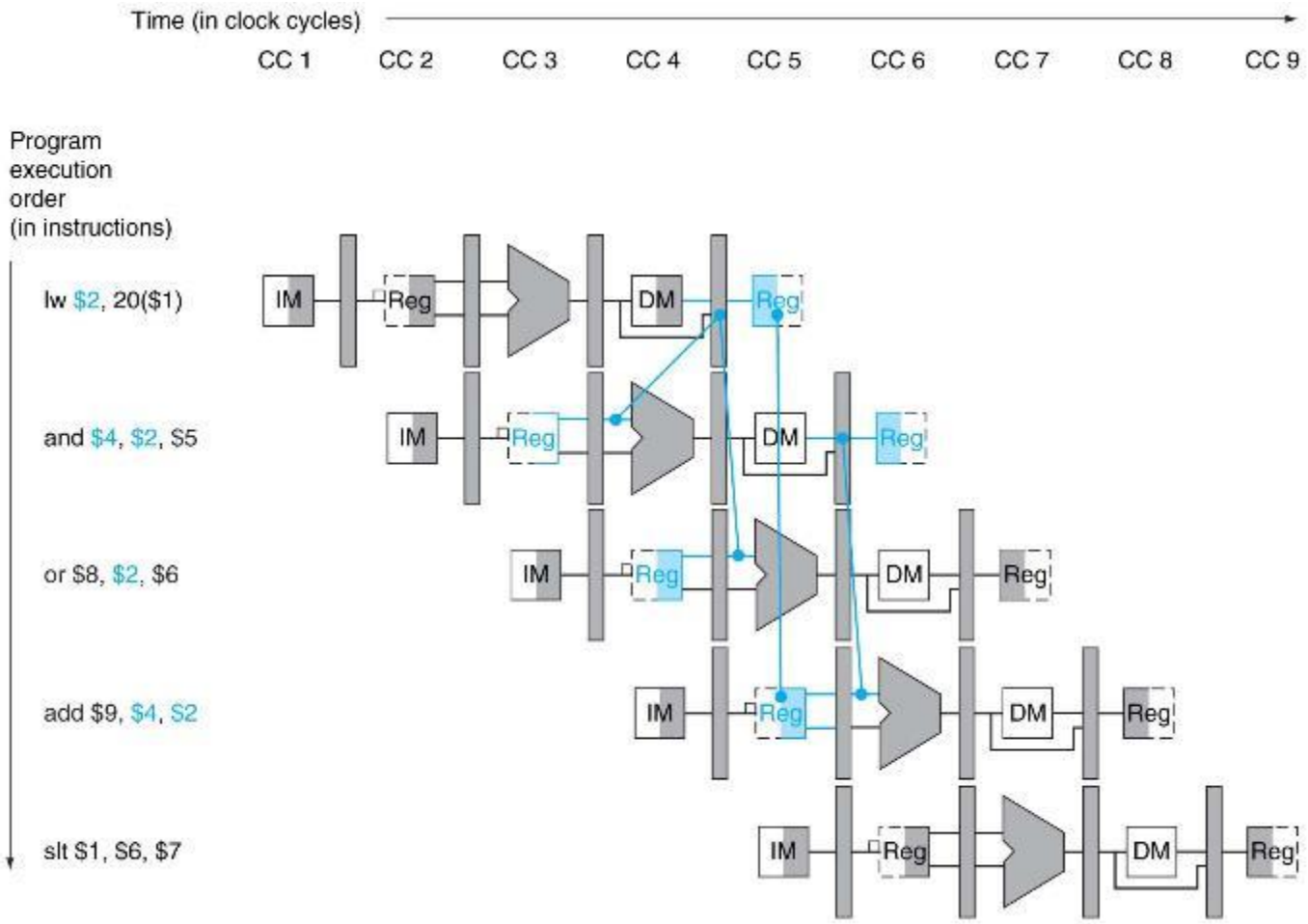


Testes realizados

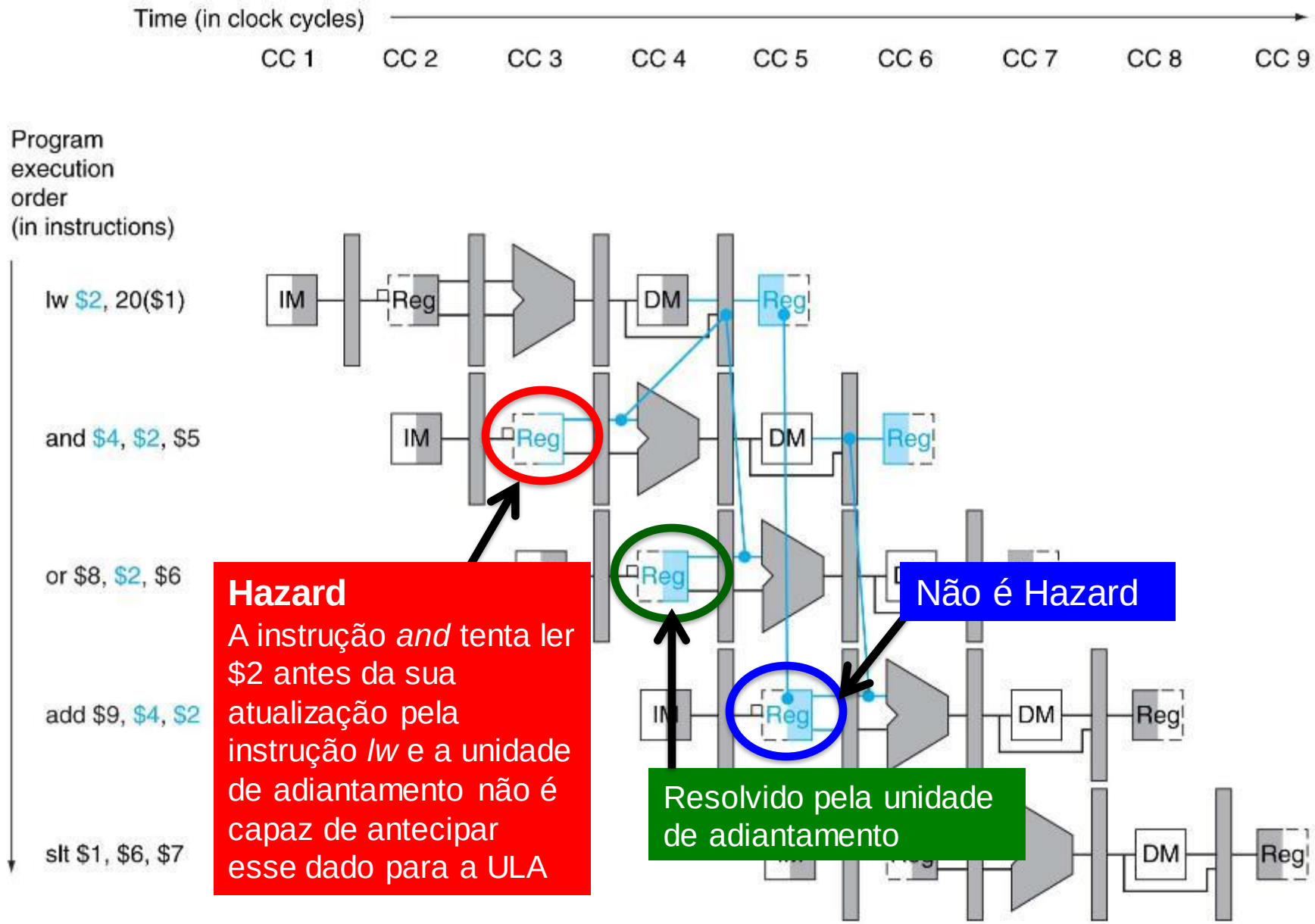
EX/MEM.Rd = ID/EX.Rs
EX/MEM.Rd = ID/EX.Rt
MEM/WB.Rd = ID/EX.Rs
MEM/WB.Rd = ID/EX.Rt

O pipeline real: conflito de dado

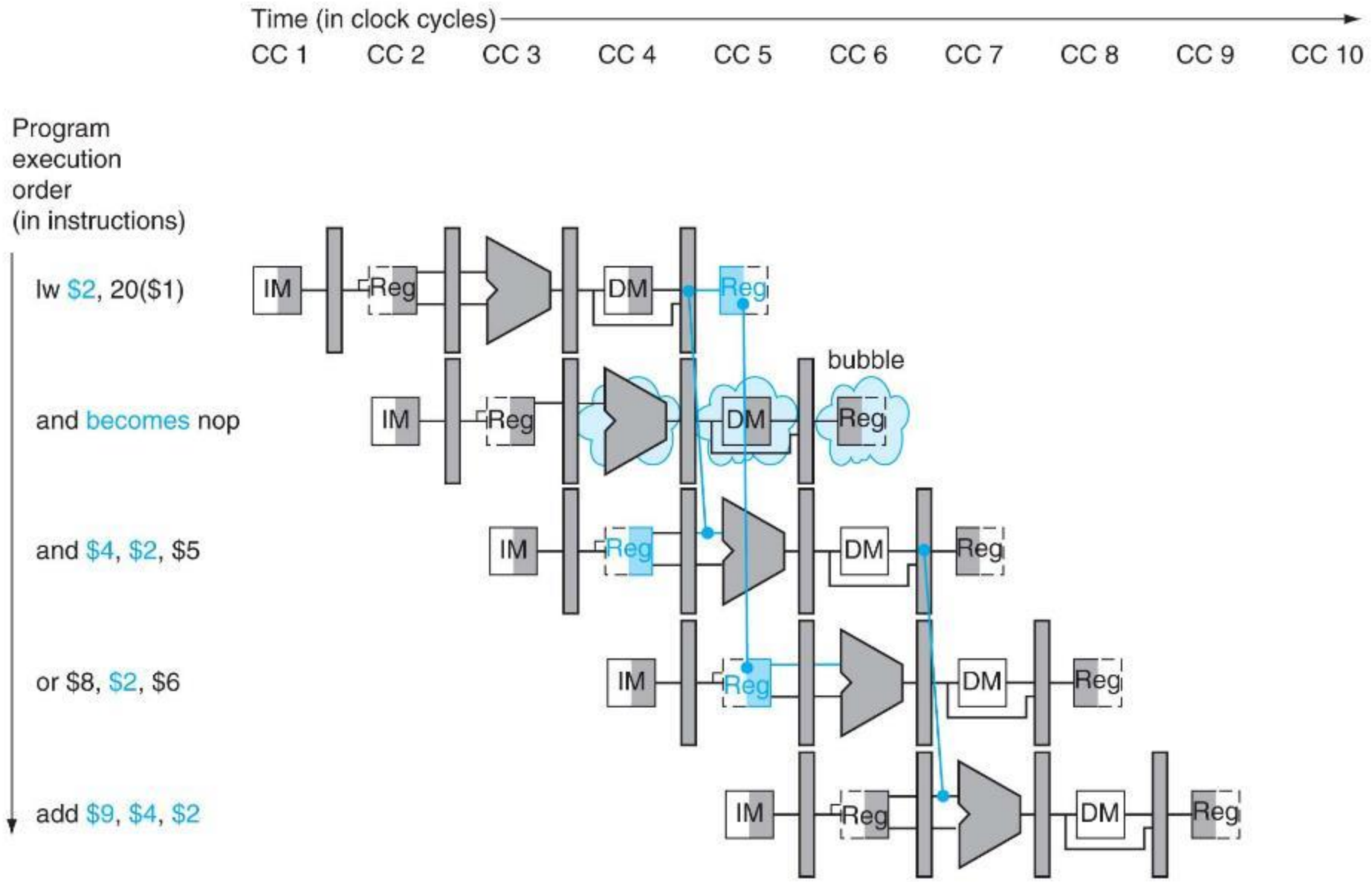
- **Inserção de bolhas (*stalls*)**
  - O adiantamento não resolve o problema de uma instrução depender do resultado de uma instrução *lw* anterior a ela



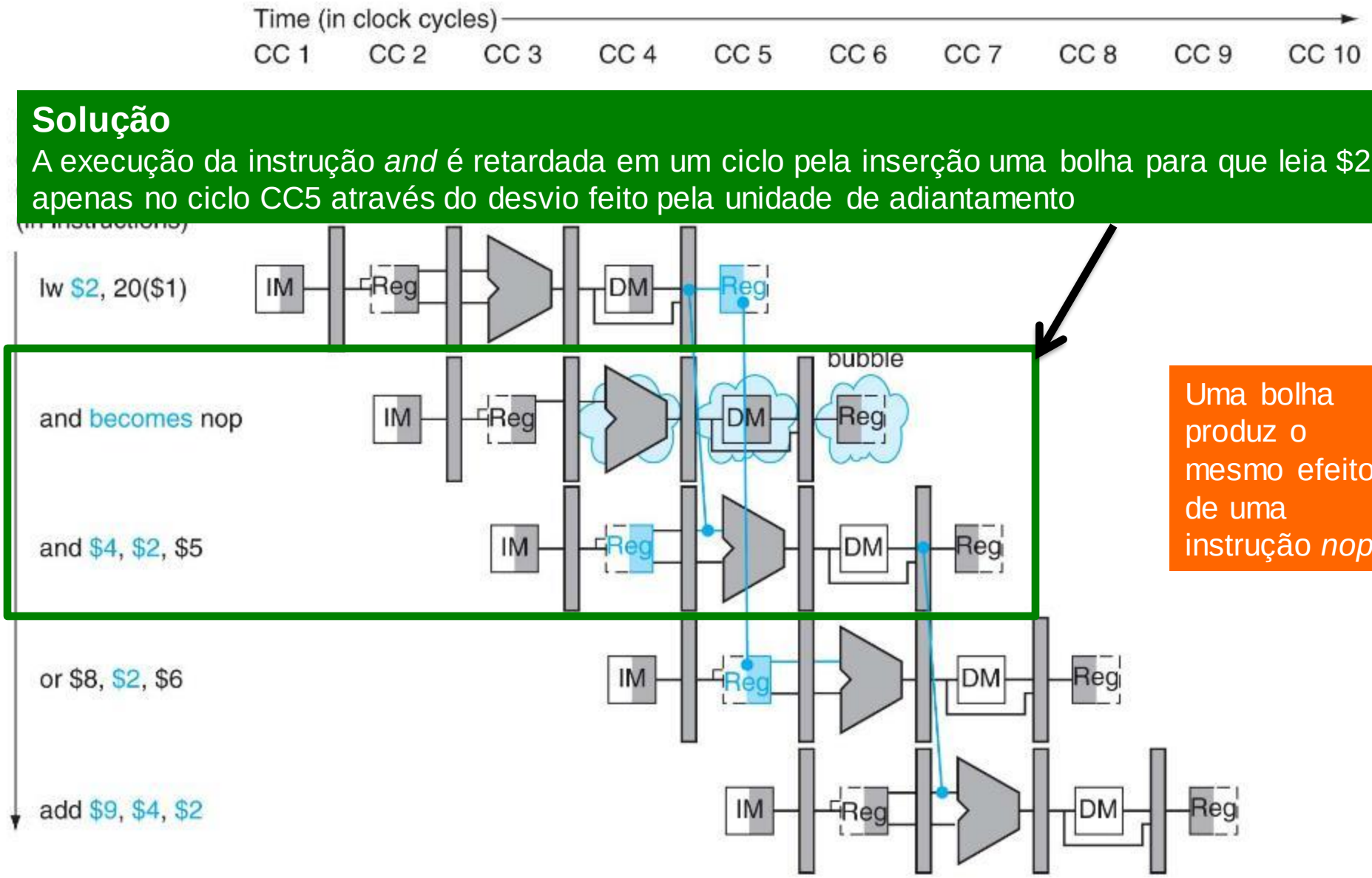
O pipeline real: conflito de dado



O pipeline real



O pipeline real: conflito de dado



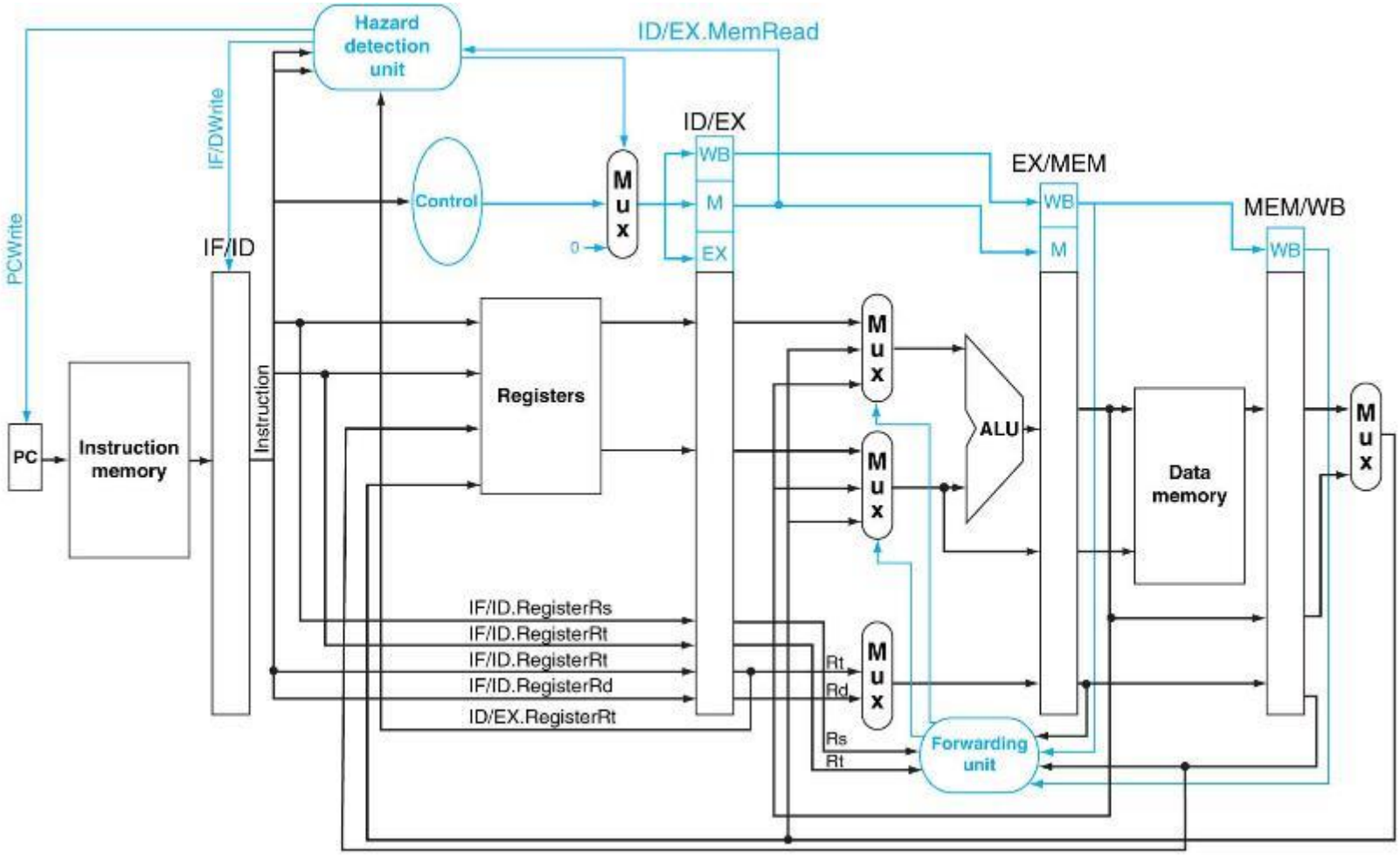
Solução

A execução da instrução *and* é retardada em um ciclo pela inserção uma bolha para que leia $2 apenas no ciclo CC5 através do desvio feito pela unidade de adiantamento

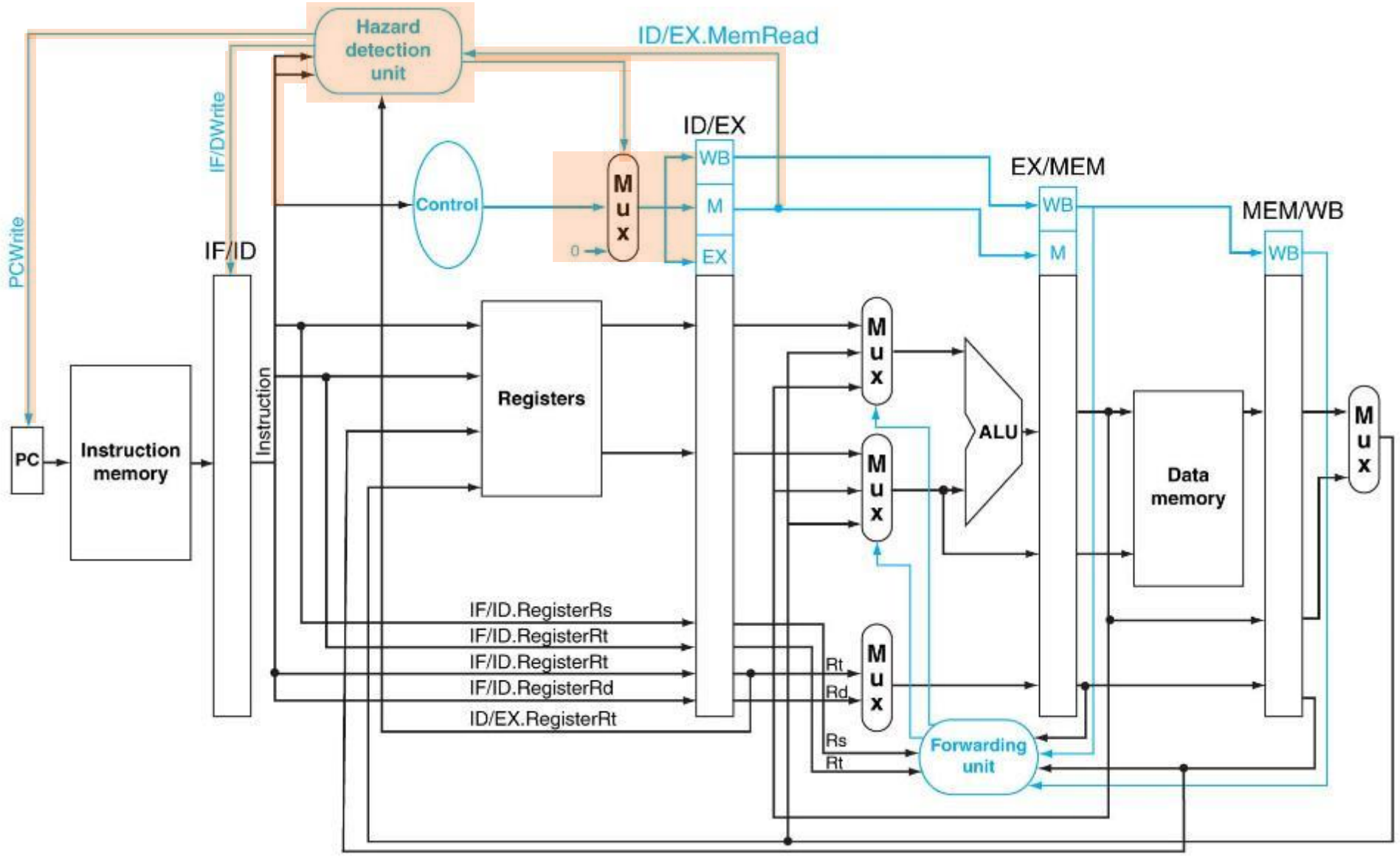
Uma bolha produz o mesmo efeito de uma instrução *nop*

O pipeline real: conflito de dado

- **Unidade de detecção de conflito (*hazard*)**
  - **Insere uma bolha no pipeline**: zera os sinais de controle e desabilita a escrita no PC e no registrador IF/ID

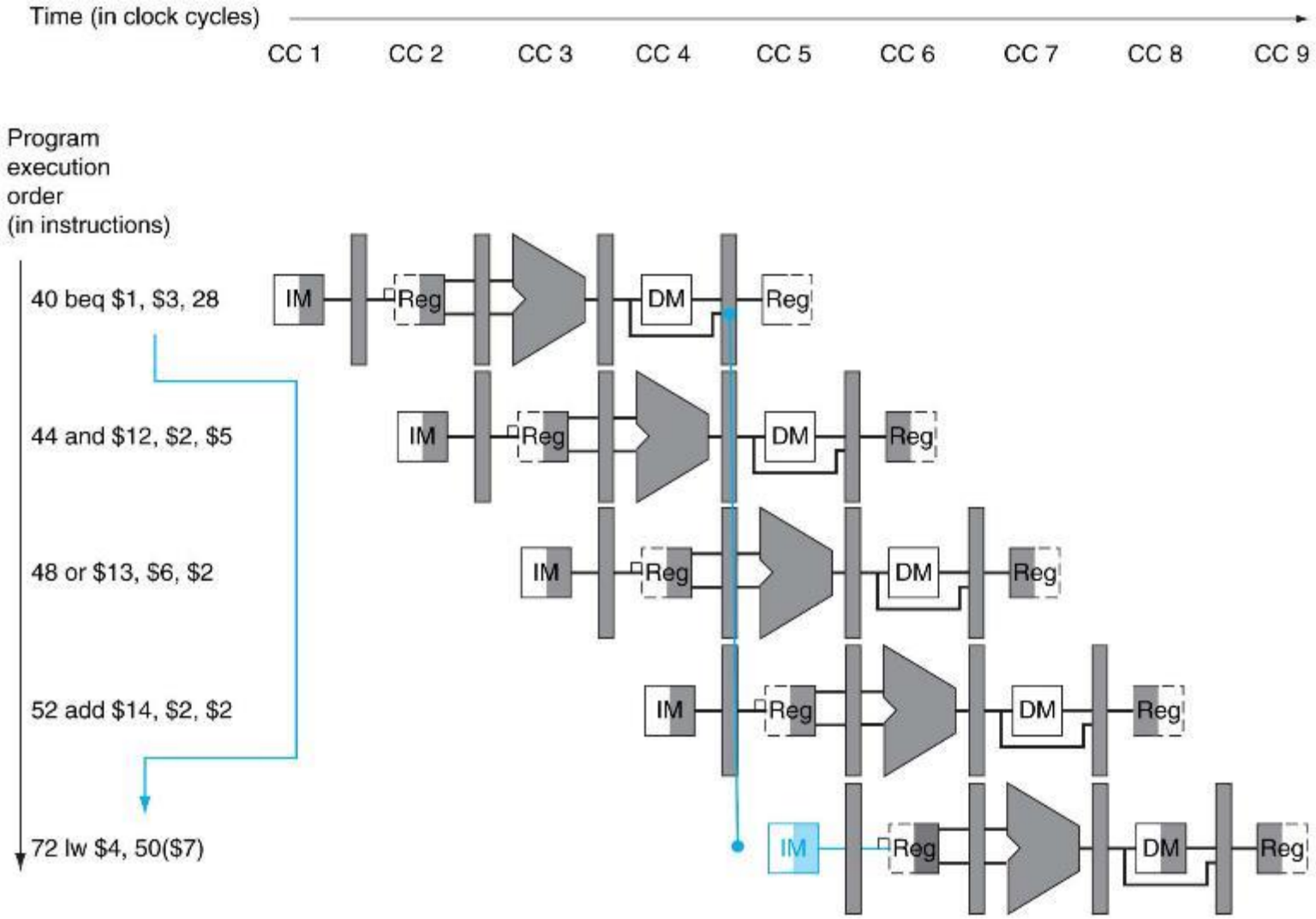


O pipeline real: conflito de dado

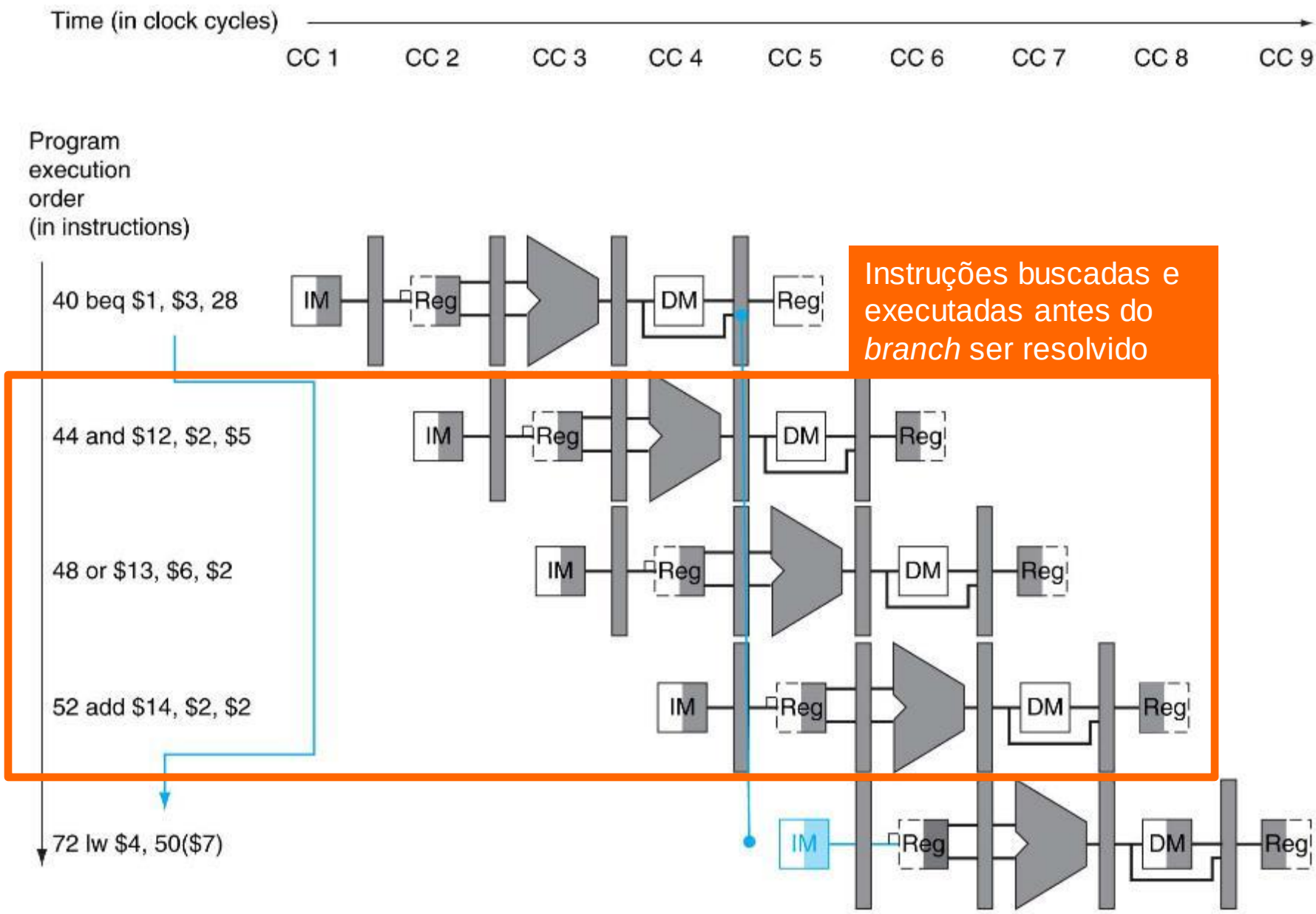


O pipeline real: conflito de controle

- Ocorre quando instruções há instruções de desvio no fluxo de instruções do pipeline



O pipeline real: conflito de controle

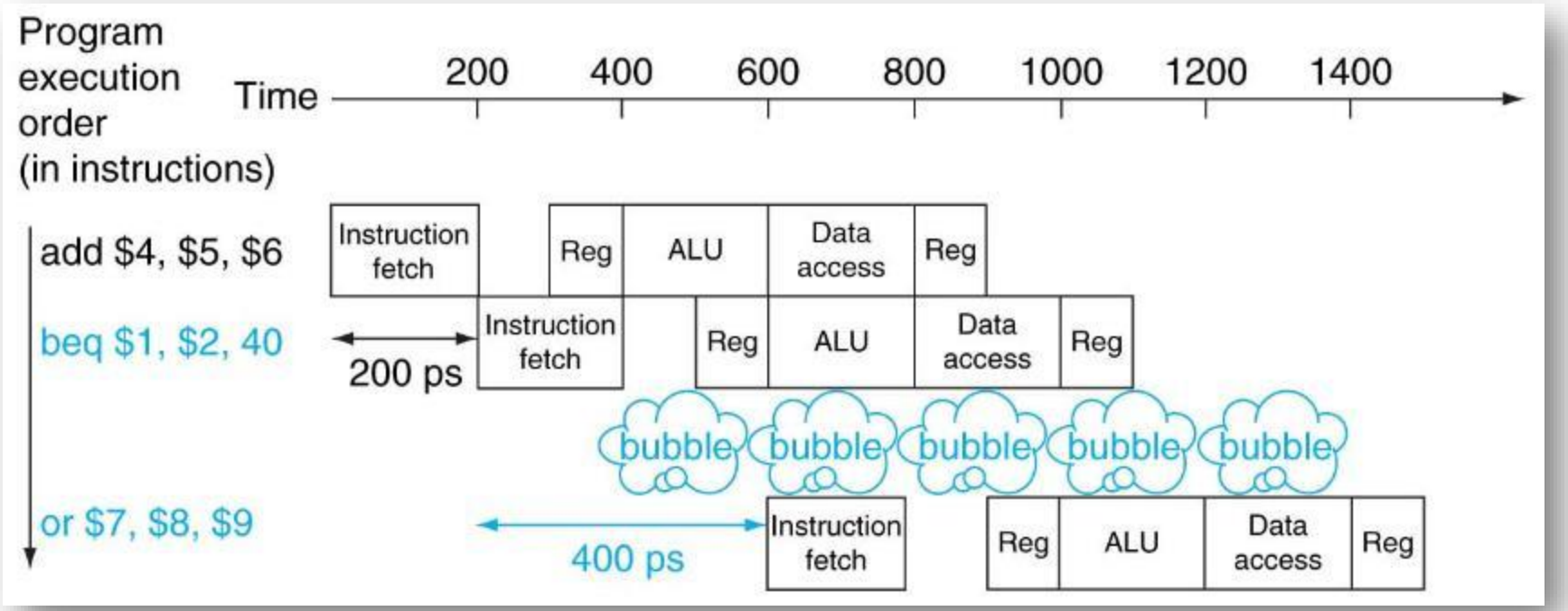


O pipeline real: conflito de controle

- **Soluções**
  1. **Congelamento do pipeline**: esperar para buscar próxima instrução até que o desvio seja executado
  2. **Execução especulativa**: Predizer o comportamento do desvio

O pipeline real: conflito de controle

- **Congelamento do pipeline (inserção de bolha)**
  - Retarda a busca da próxima instrução até que o desvio seja executado



  - Solução simples, mas com perda de desempenho

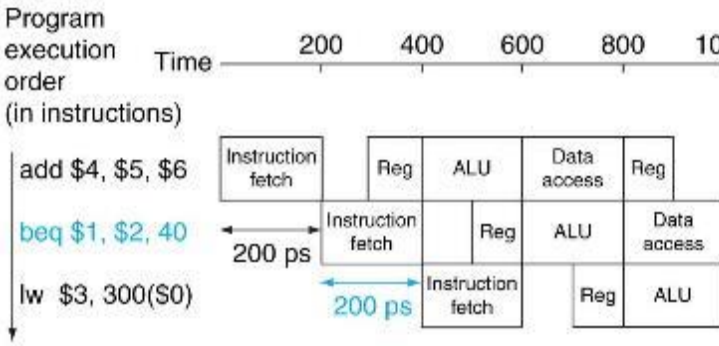
O pipeline real: conflito de controle

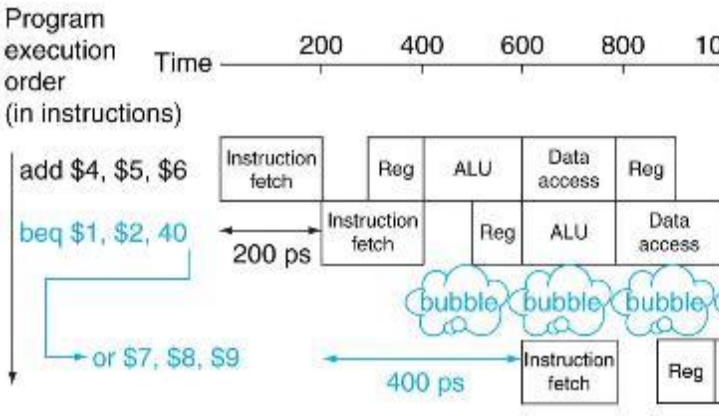
- **Execução especulativa**
  - Utiliza técnicas de predição em que assume a execução de um dos ramos do desvio
- **Dois tipos**
  - Predição estática
  - Predição dinâmica

O pipeline real: conflito de controle

- **Predição estática simples**
  - Assume sempre a mesma decisão para os desvios condicionais:
    - Desvio nunca será tomado
      ou
    - Desvio sempre será tomado (desvio incondicional)
  - Se acertar, o pipeline prossegue com velocidade máxima
  - Se errar, será necessário atrasar o pipeline

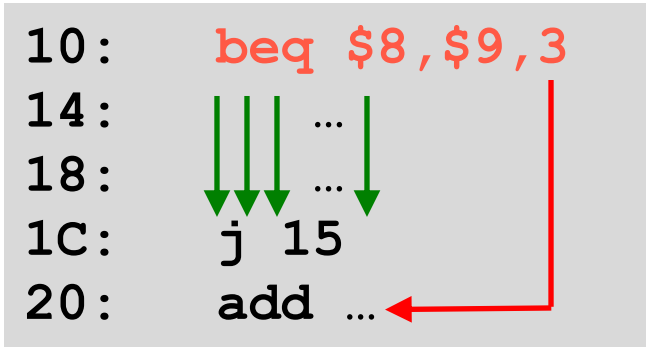
Exemplo: Predição estática que assume que os desvios não serão tomados





O pipeline real: conflito de controle

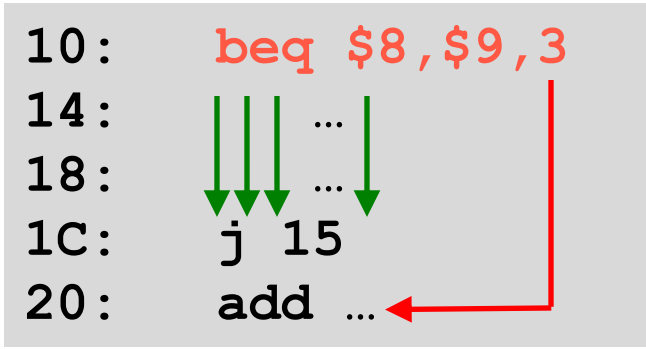
- **Predição estática sofisticada**
  - Assume que parte dos desvios ocorrem e outros não ocorrem
    - Para endereços menores (*backward*) são sempre tomados
    - Para endereços maiores (*forward*) não são tomados
  - Melhora o desempenho da execução de laços de repetição
    - **Laços dos tipos *for* e *while***
      - Possuem um desvio condicional no início para um endereço maior e que só é tomado uma vez na última iteração

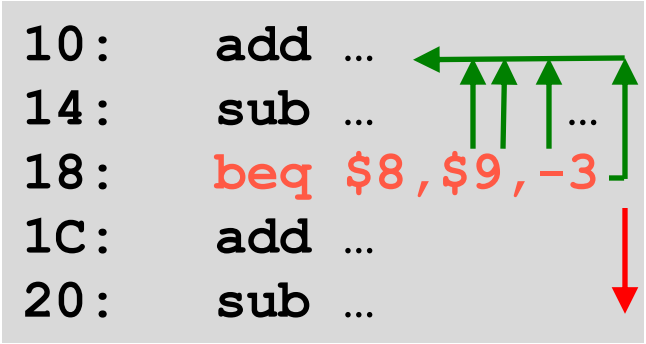


Nota: No *branch*, o operando imediato informa a distância (em instruções) até a instrução para a qual será feito o desvio, considerando que o PC já está apontando para a instrução seguinte (PC = endereço da instrução atual + 4).

O pipeline real: conflito de controle

- **Predição estática sofisticada**
  - Assume que parte dos desvios ocorrem e outros não ocorrem
    - Para endereços menores (*backward*) são sempre tomados
    - Para endereços maiores (*forward*) não são tomados
  - Melhora o desempenho da execução de laços de repetição
    - **Laços dos tipos *for* e *while***
      - Possuem um desvio condicional no início para um endereço maior e que só é tomado uma vez na última iteração
    - **Laços dos tipos *do-while***
      - Possuem um desvio condicional no final para um endereço menor e que é tomado em todas as iterações, menos na última





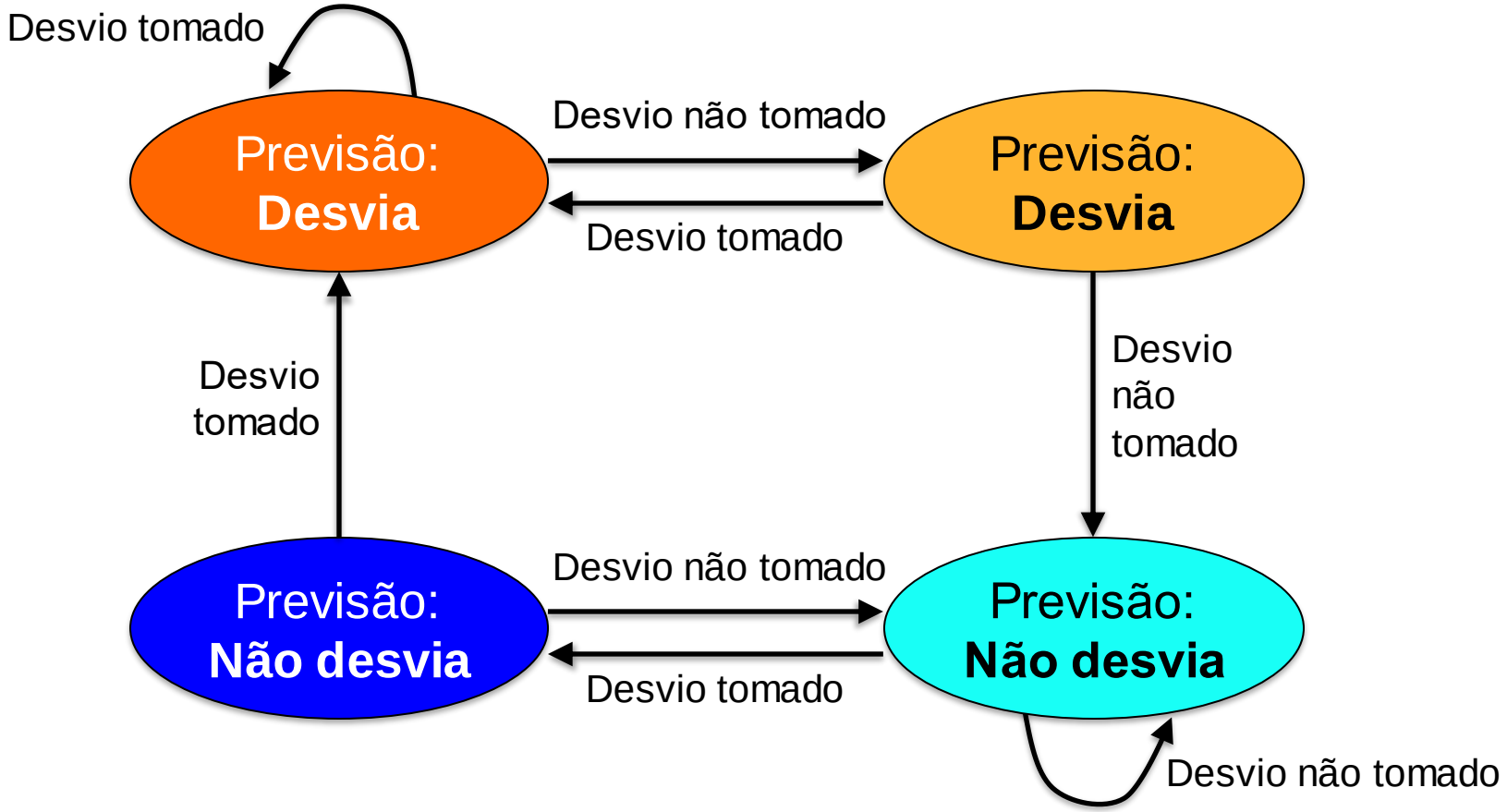
Nota: No *branch*, o operando imediato informa a distância (em instruções) até a instrução para a qual será feito o desvio, considerando que o PC já está apontando para a instrução seguinte (PC = endereço da instrução atual + 4).

O pipeline real: conflito de controle

- **Predição dinâmica**
  - Analisa o histórico de cada desvio e pode mudar a predição para um mesmo ponto de desvio com o decorrer da execução do programa em função dos erros de predição
  - Soluções
    - Chave desvia/não desvia
    - Desvio retardado (*delayed branch*)

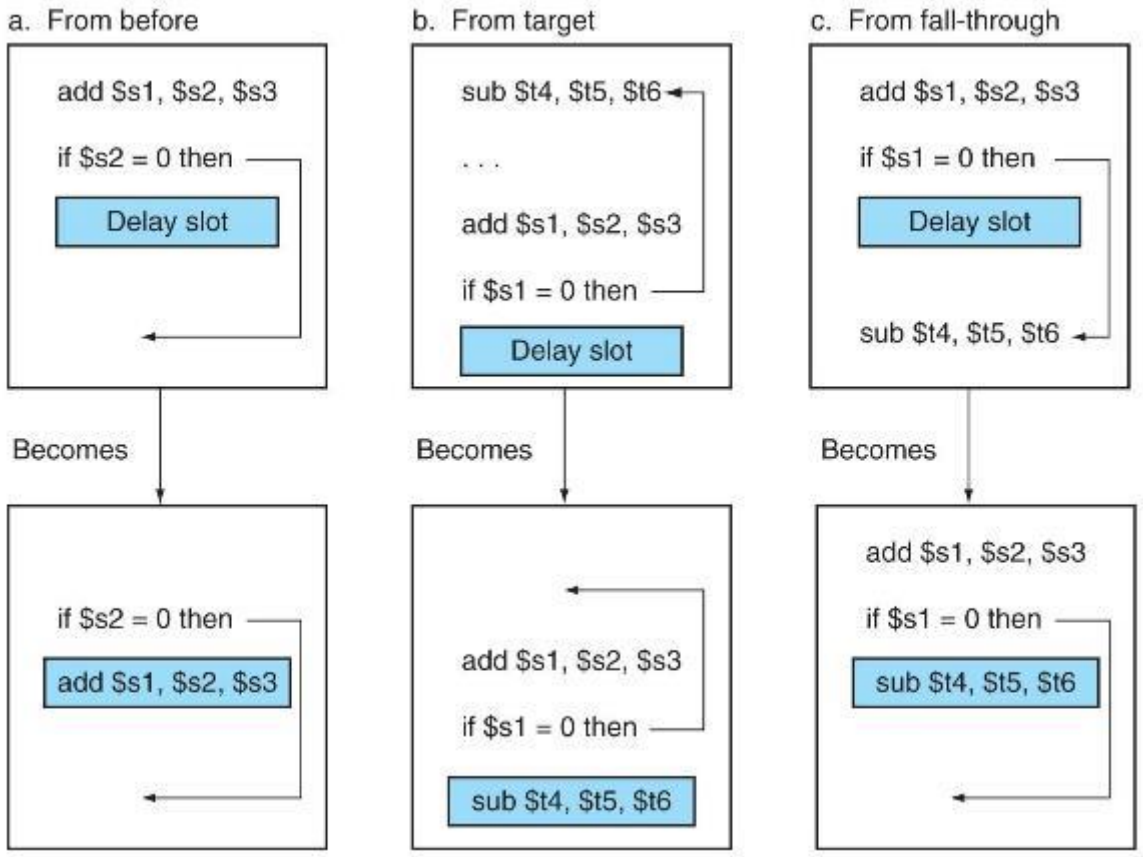
O pipeline real: conflito de controle

- **Chave desvia/não desvia**
  - Máquina de estados que decide antecipadamente tomar (ou não tomar) um desvio e muda seu posicionamento caso erre



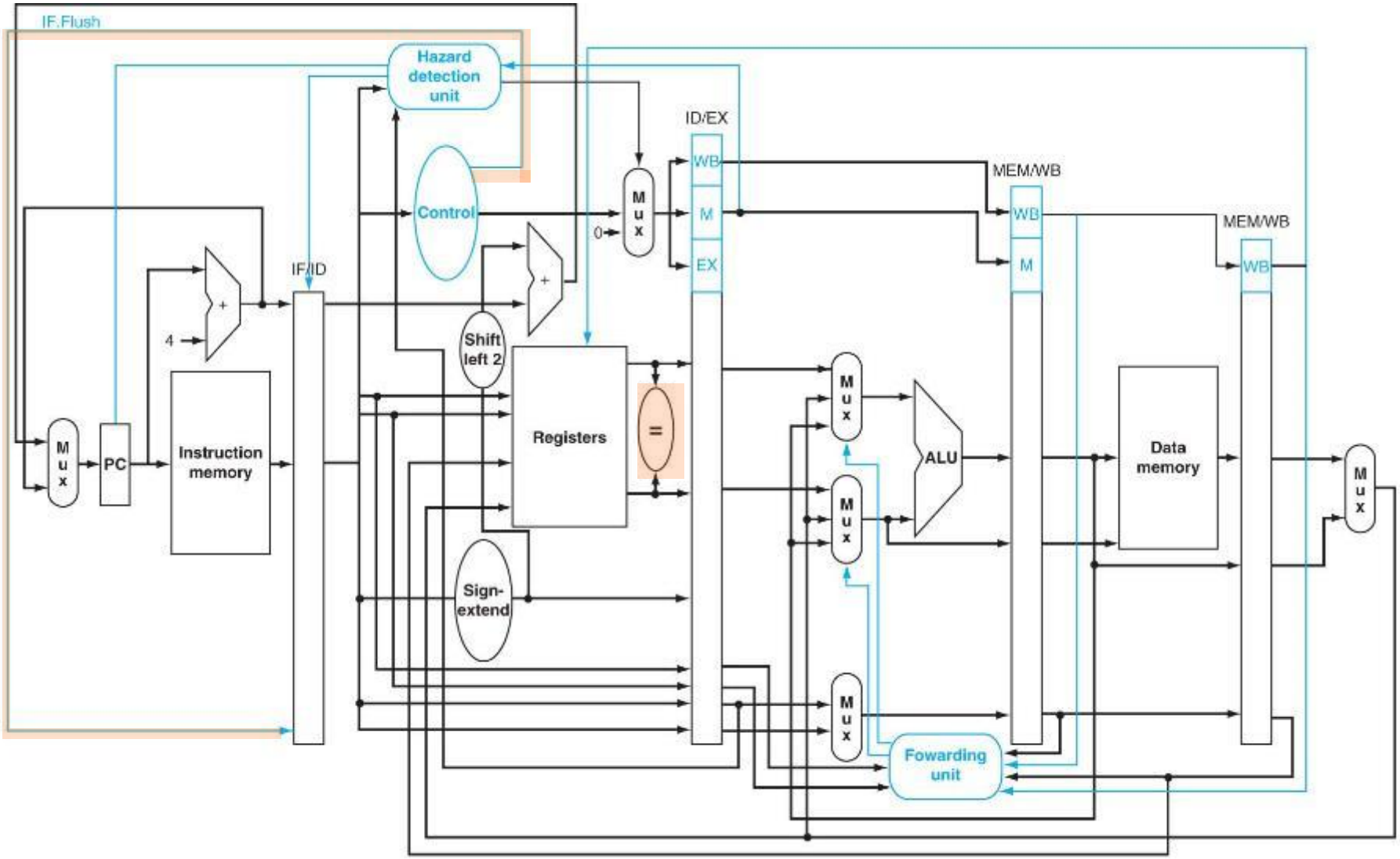
O pipeline real: conflito de controle

- **Desvio retardado (*delayed branch*)**
  - Coloca uma instrução "útil" para ser executada em um *slot* logo após a instrução de desvio, de modo a manter o pipeline preenchido, evitando a parada (tarefa do compilador)



O pipeline real: conflito de controle

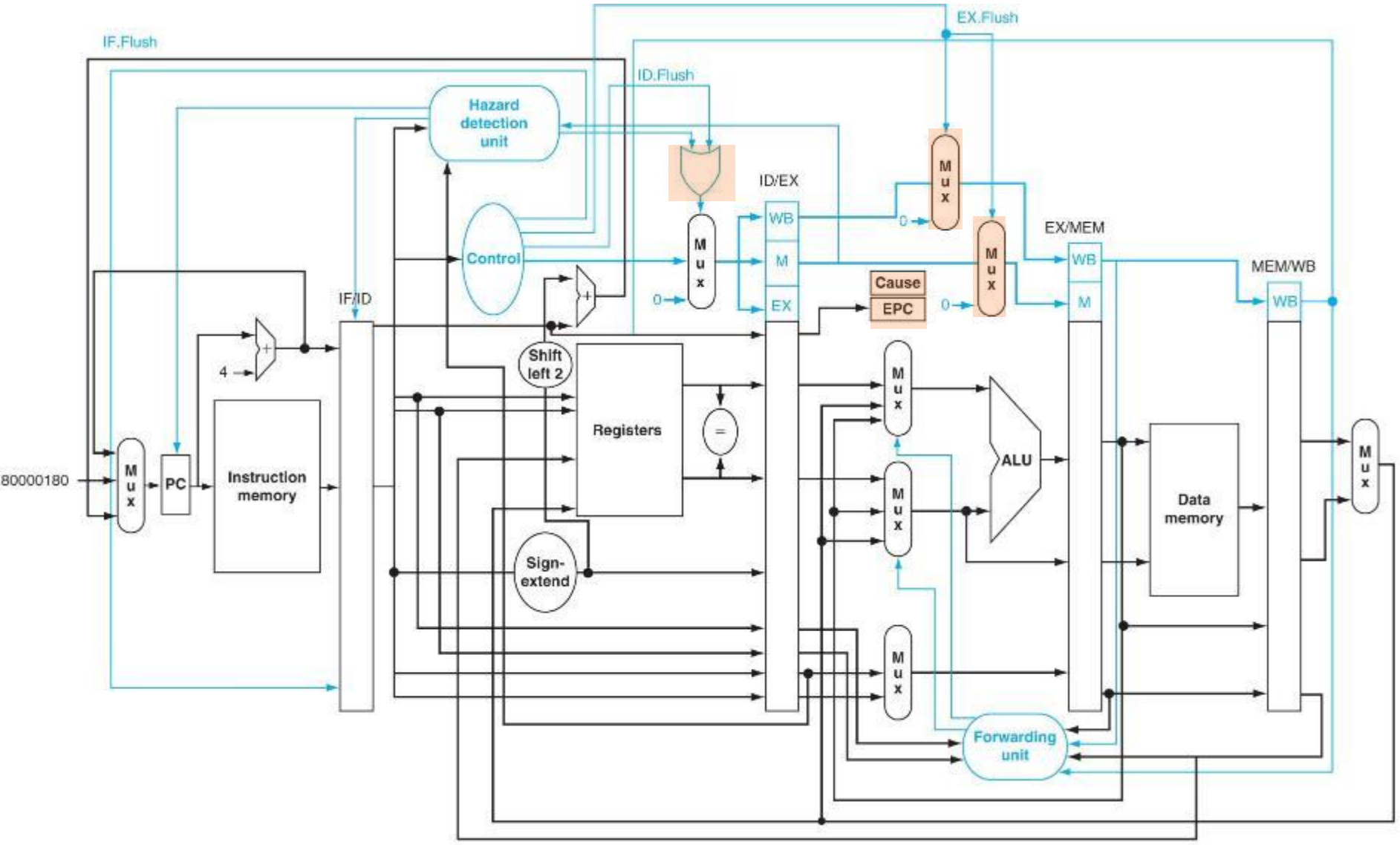
- Modificações no caminho de dados



Tratamento de exceções

- **Exceção:** evento que interrompe a execução do programa
- **Tipos**
  - Opcode não válido
  - Overflow
  - Mal funcionamento do hardware
  - Chamada do S.O
  - Requisição de Dispositivo de E/S
- **Comportamento do pipeline**
  - Instruções antes da que causou exceção devem ser terminadas
  - Instruções após a que causou exceção devem ser anuladas (*flushing*)
  - Registradores EPC e Cause para identificar a origem da exceção

Tratamento de exceções



Resumo

- A técnica de pipelining permite aumentar o desempenho do processador pela exploração do paralelismo em nível de instruções
- Várias instruções são executadas ao mesmo tempo, mas em etapas diferentes do ciclo de instrução
- O pipelining não reduz o tempo para executar cada instrução, mas aumenta a taxa de instruções executadas no tempo

Resumo

- A implementação do pipeline no RISC-V baseia-se na organização do RISC-V monociclo, com a inclusão de registradores entre os estágios que executam as etapas do ciclo de instrução
- Em função de conflitos estruturais, de dado e de controle, é necessário replicar recursos e implementar unidades para detectar e resolver conflitos
- Os recursos adicionais tornam o RISC-V pipeline mais caro que os demais, mas resulta em um processador com maior desempenho