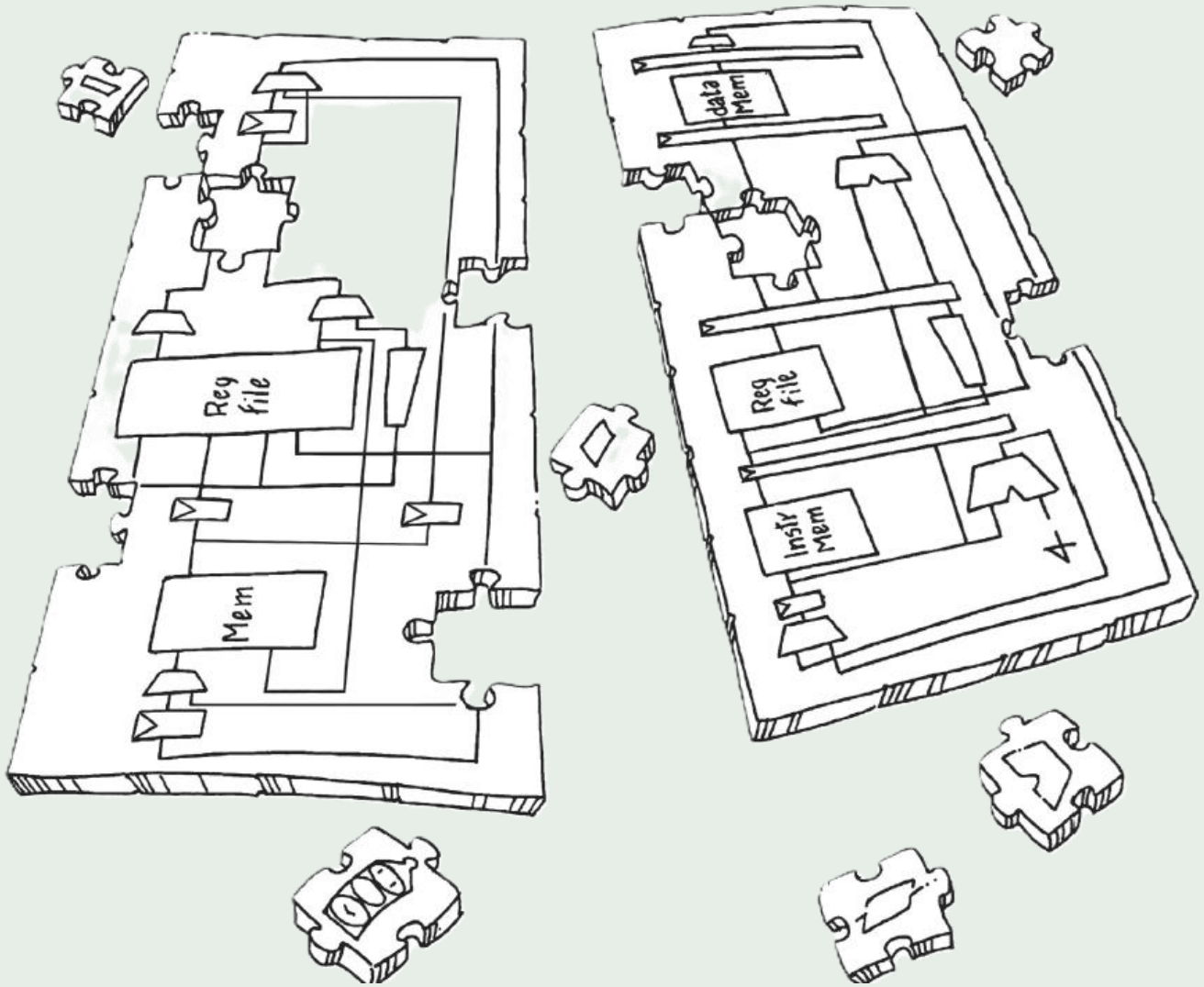
Departamento de Computação - UFSCar



Fonte: Adaptado de Digital Design and Computer Architecture RISC-V Edition Sarah L. Harris David Harris

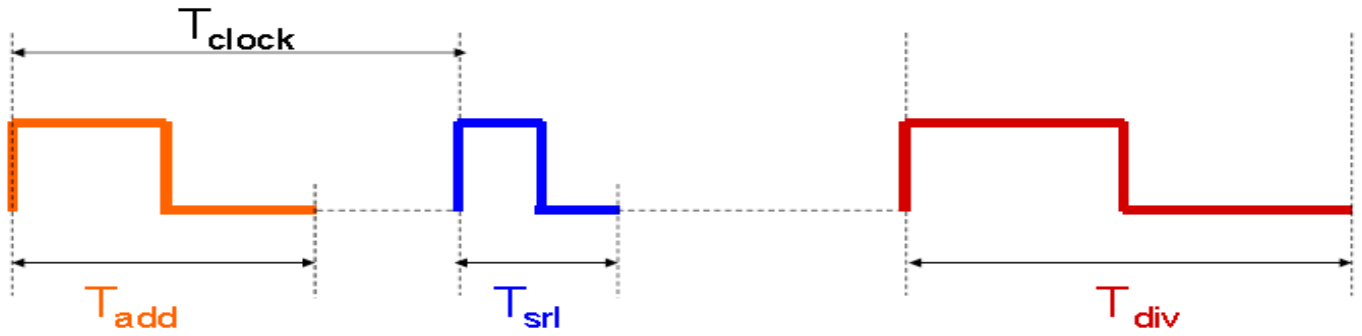
Arquitetura e Organização de Computadores 1

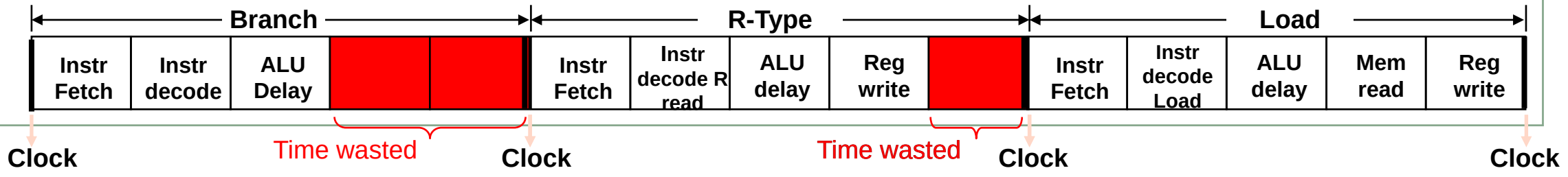
Prof. Luciano de Oliveira Neris
luciano@dc.ufscar.br

Datapath e Controle Pipelined para RISC-V

Datapath Monociclo

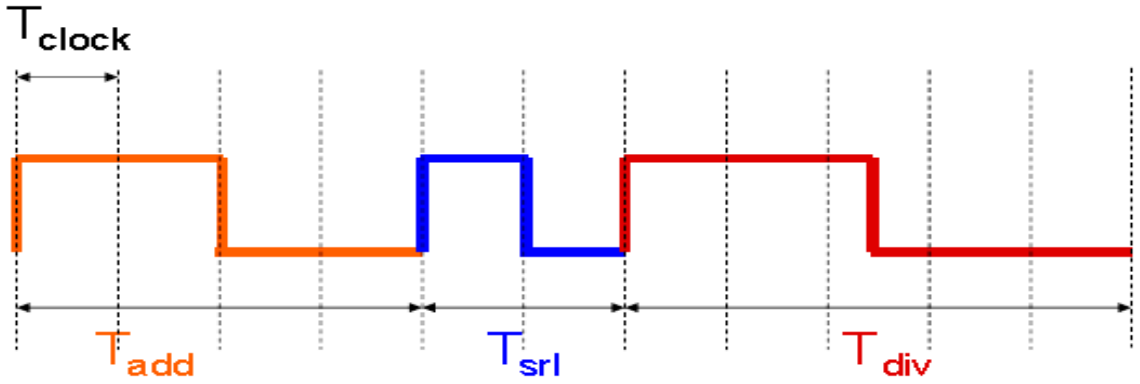
- Cada instrução é executada em um 1 ciclo de clock
- Ciclo de clock deve ser longo o suficiente para executar a instrução mais longa
- Desvantagem: velocidade global limitada à velocidade da instrução mais lenta

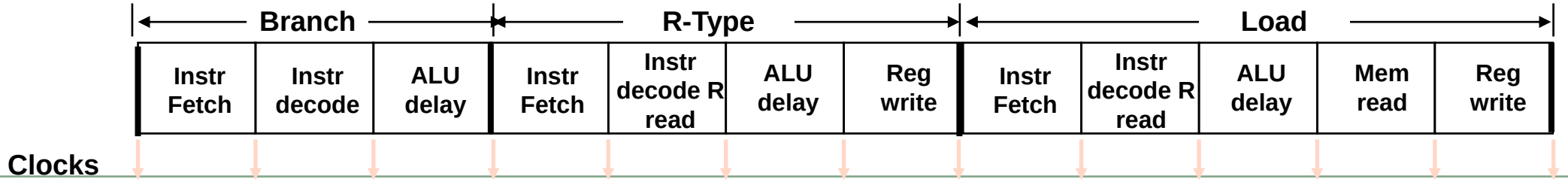




Datapath Multi-ciclo

- Quebra o ciclo de execução em vários passos
- Executa cada passo em um ciclo de clock
- Cada instrução usa apenas o número de ciclos que ela necessita

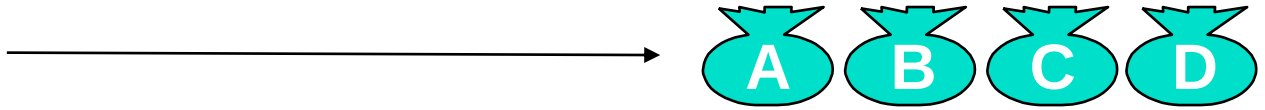




Pipeline

Lavanderia

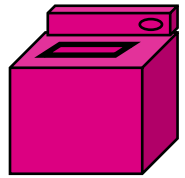
4 Sacolas de roupa suja - A,B,C,D



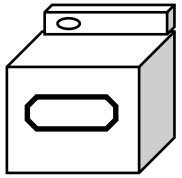
Tarefa p/ cada uma delas:

- lavar, secar, passar, guardar

° **Lavar gasta 30 minutos**



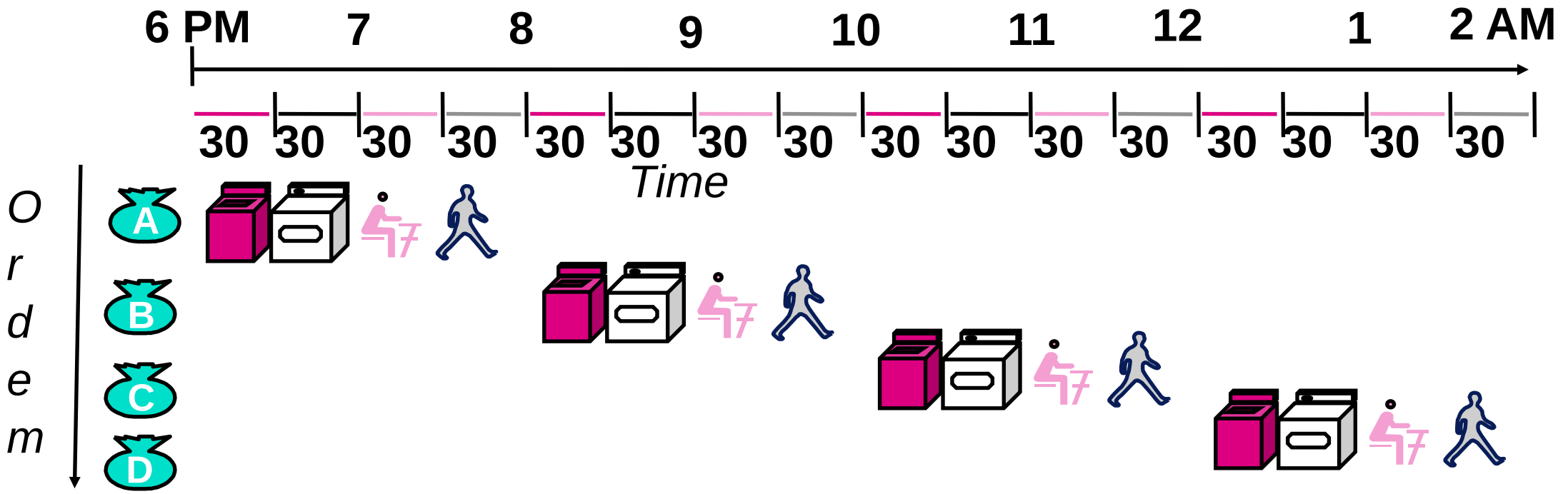
° **Secar gasta 30 minutos**



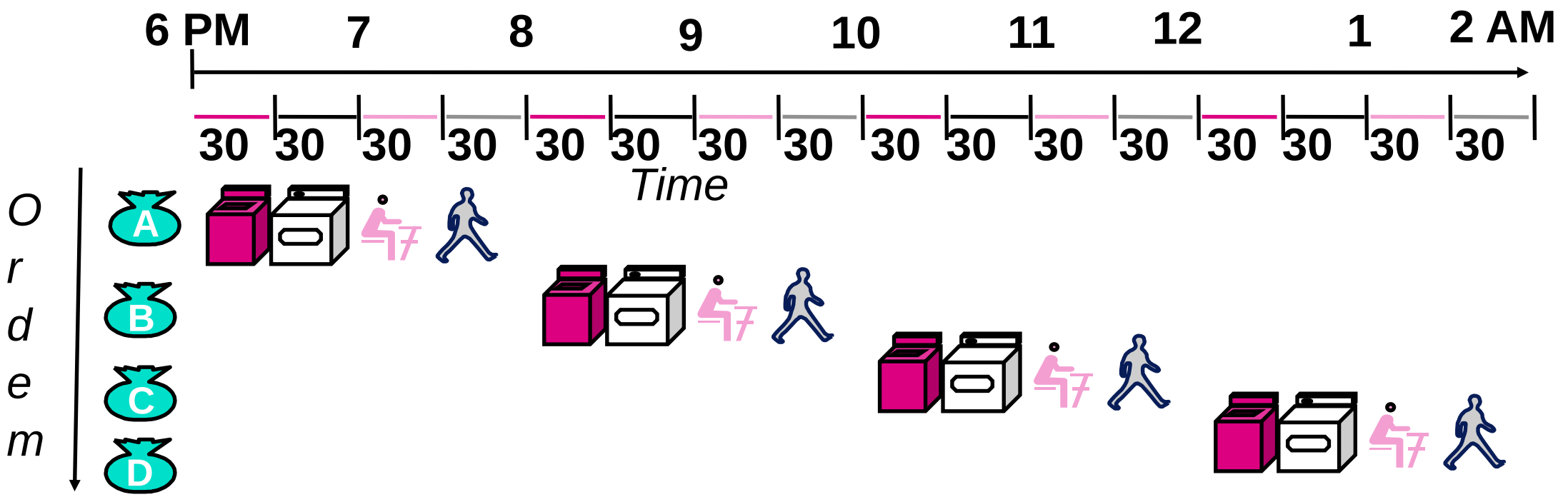
° **Passar gasta 30 minutos**

° **Guardar gasta 30 minutos**

Lavanderia Sequencial

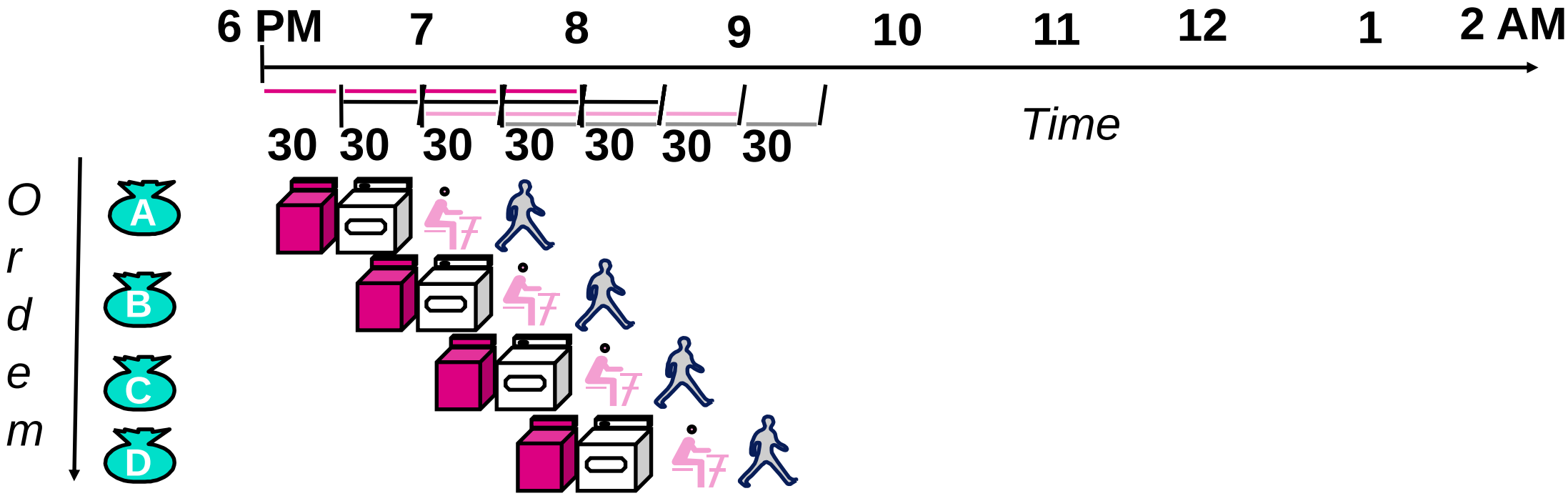


Lavanderia Sequencial

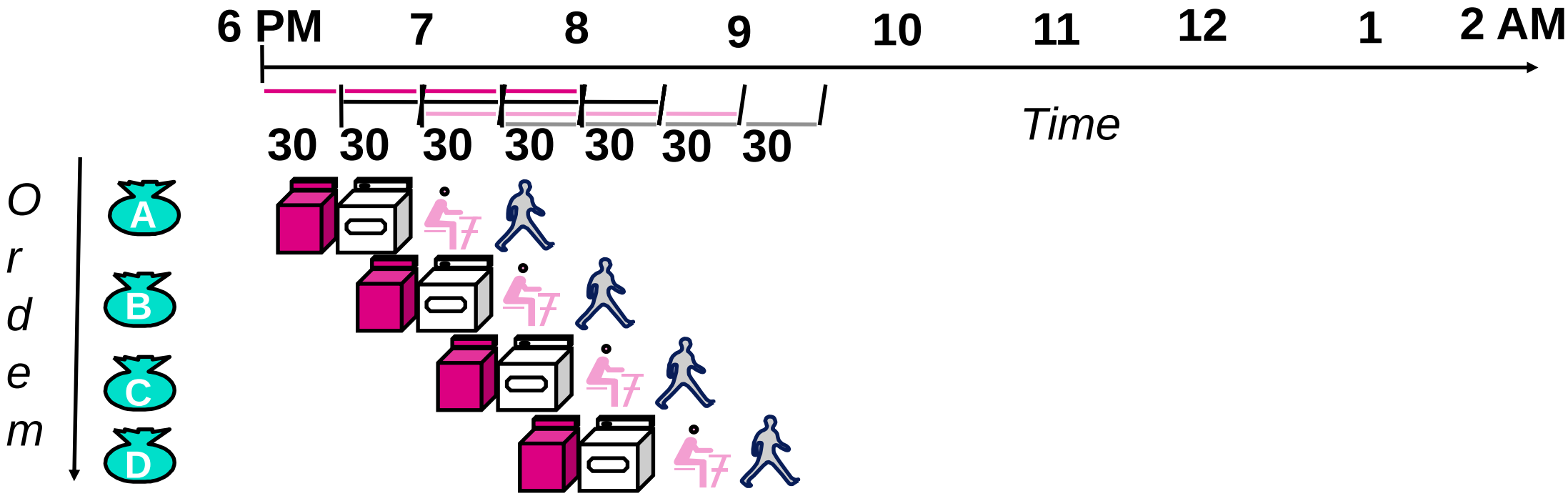


Lavanderia sequencial gasta 8 horas p/ 4 cargas de roupas

Lavanderia em Pipeline

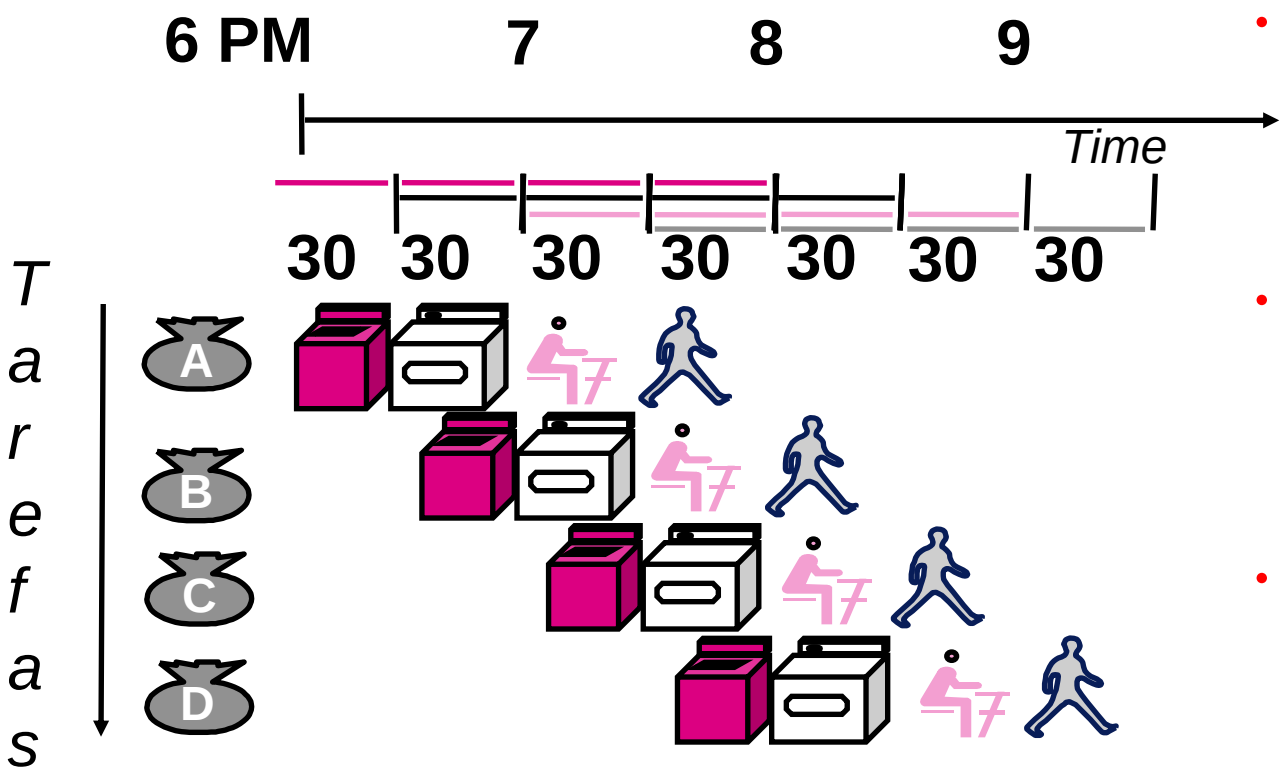


Lavanderia em Pipeline



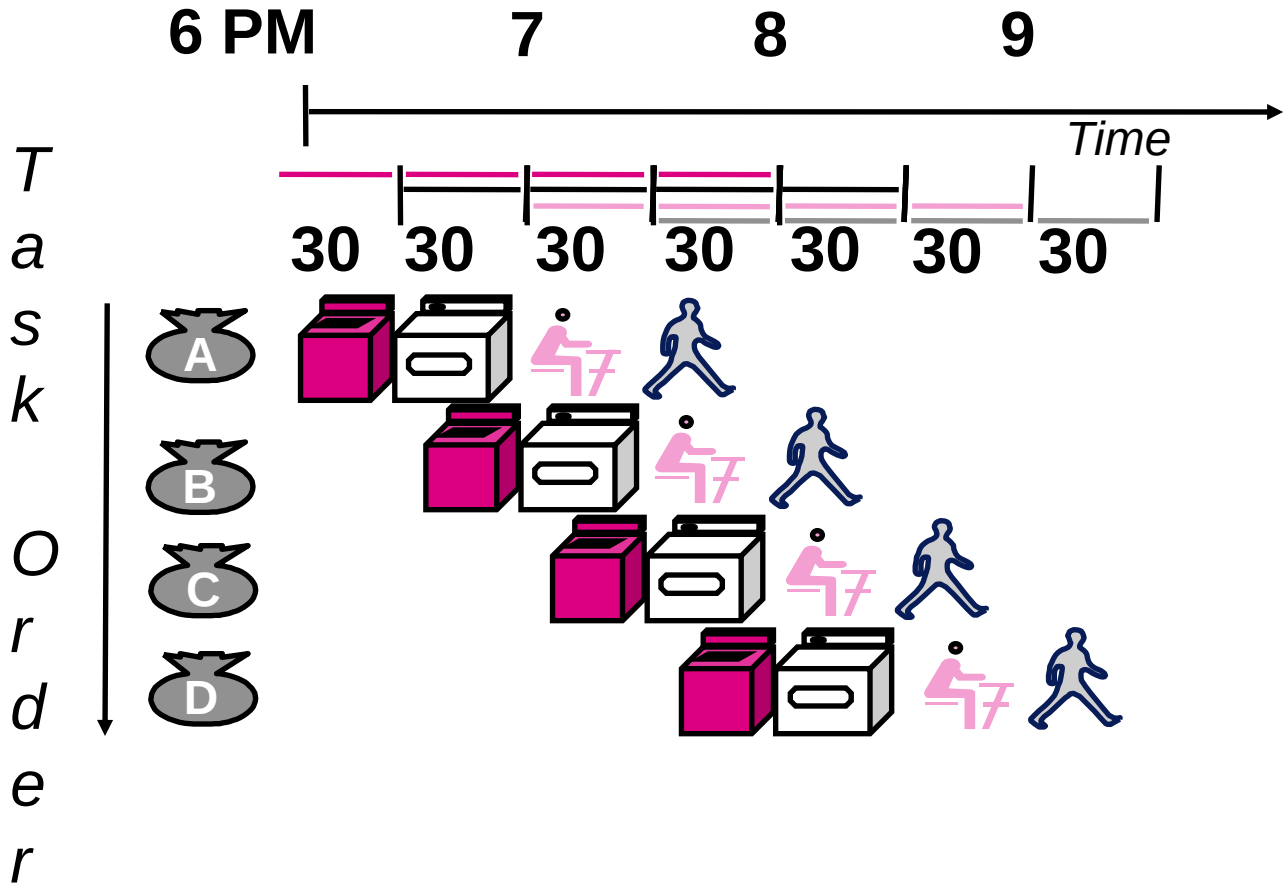
Lavanderia em pipeline gasta 3.5 horas p/ 4 cargas de roupas

Pipeline: Conclusões (1/2)



- Pipelining não melhora a latência de uma única tarefa, mas sim a produção do trabalho completo.
- Multiplas tarefas são executadas simultaneamente, usando recursos diferentes.
- Speedup Potencial: = Número de stágios do pipeline

Pipeline: Conclusões (2/2)



- Suponha lavar passe a gastar apenas 20 minutos. Quanto melhoraria o pipeline completo?
- Nada – a taxa do pipeline é limitada pelo estágio mais lento.
- O tempo de execução de cada estágio deve ser balanceado.

Pipelining

- Tecnologia chave p/ alcançar alto desempenho

- Pipelining é uma técnica que permite a intercalação de estágios diferentes na execução de várias instruções simultaneamente.
  - O hardware processa mais de uma instrução de cada vez, sem aguardar o término de uma instrução para iniciar a próxima instrução.

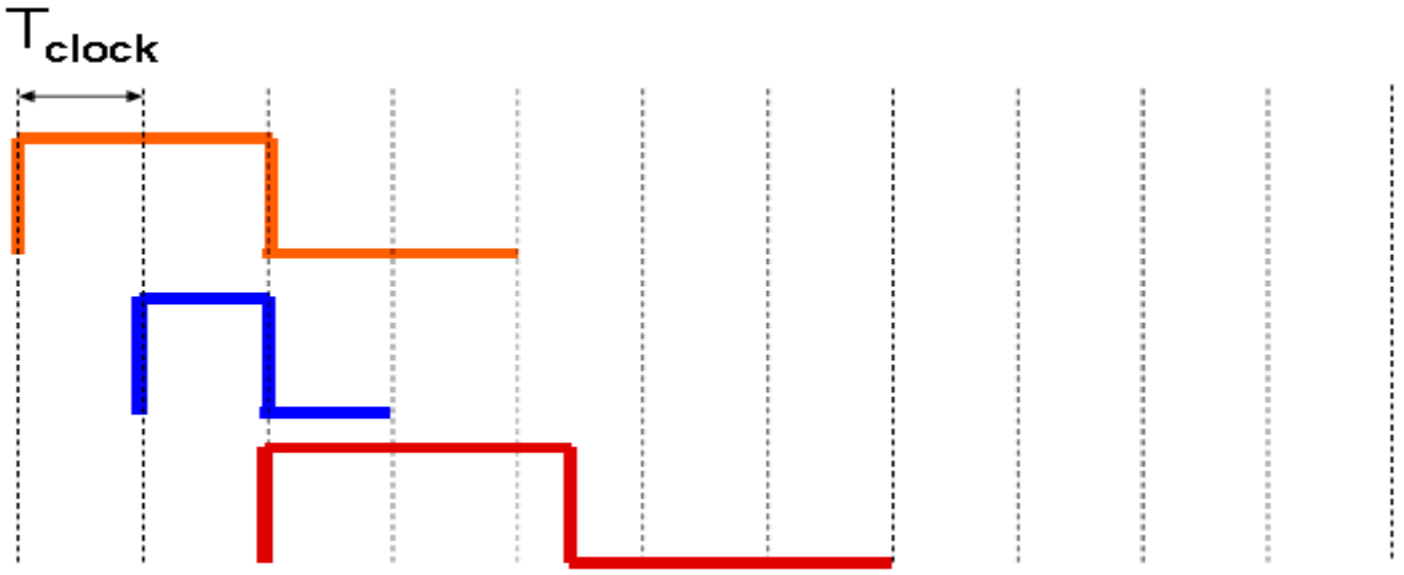
- Possível devido aos seguintes fatos:
  - A execução de uma instrução pode ser subdividida em vários estágios
  - Duas ou mais instruções podem estar em estágios diferentes

Pipelining

- Pode ser visto como uma "linha de montagem", constituída por vários passos (estágios);
- Cada estágio é conectado ao seguinte, análogo a uma canalização (cano= pipe)
- Instruções entram no início da canalização (pipelining), e saem no final

Arquitetura tipo Pipelined

- Cada instrução é executada em múltiplos ciclos
- Executa uma etapa de cada instrução em cada ciclo
- Processa múltiplas instruções em paralelo



Desempenho do Pipelining

- Pipelining reduz o tempo médio na execução de um conjunto de instruções → Diminui o CPI médio
- Pipelining permite a implementação de estágios mais curtos, diminuindo o tempo de ciclo da máquina → Aumenta o clock rate.

$$\text{CPU time} = \frac{\text{IC} \times \text{CPI}}{Clockrate}$$

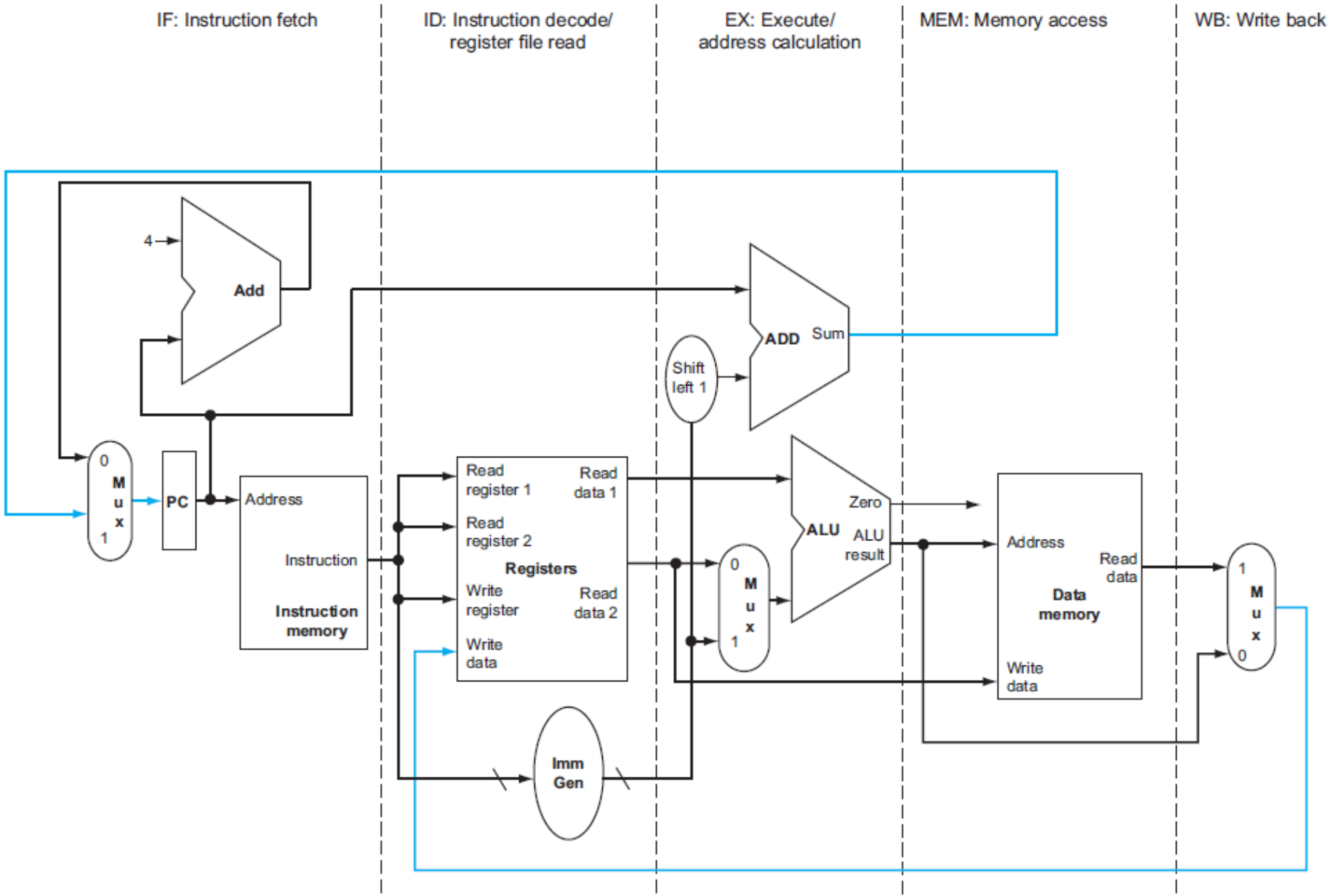
- O uso do Pipelining é totalmente transparente ao programador/compilador → Técnica altamente bem sucedida!

Pipelining para RISC-V

- Arquiteturas RISC (ex: RISC-V) se adequam muito bem à organização em pipelining, devido principalmente a:
  - Predominância de operandos do tipo registrador
  - Poucos formatos de instrução, codificação fixa (ex: 32 bits)
- Podemos estender a organização monociclo para operar no modo pipelining.
- Isso pode ser feito acrescentando-se registadores temporários entre os diversos estágios do pipeline.

Pipelining para RISC-V

- Monociclo



Pipelining para RISC-V

- Solução clássica p/ RISC-V: 5 estágios de 1 ciclo cada
  - Instruction fetch (IF) (busca)
  - Instruction decode (ID) (decodificação
  - Execution (EX) (execução)
  - Memory access (MEM) (acesso à memória)
  - Write back (WB) (escrita no banco registradores)

IF: Instruction Fetch

- IF: Busca a instrução da memória de instrução, armazenando no Registrador de Instrução (IR), atualizando o PC p/ PC+4.

ID: Instruction Decode

- ID: Envia o OPCODE p/ a unidade de controle, lê o valor dos registradores determinados na instrução (quando houver) e realiza a extensão de sinal.

EX: Execution

- EX: Se instrução = load ou store
  - ALU calcula o endereço de memória a ser acessado
- EX: Se instrução = R-type (add, sub, etc)
  - ALU executa operação c/ os registradores lidos no estágio ID
- EX: Se instrução = branch (beq)
  - ALU efetua o teste (=) entre os dois registradores lidos no estágio ID
- EX: Se instrução = jal
  - PC é substituído pelo endereço do desvio
- Além disso, vários sinais de controle são ativados

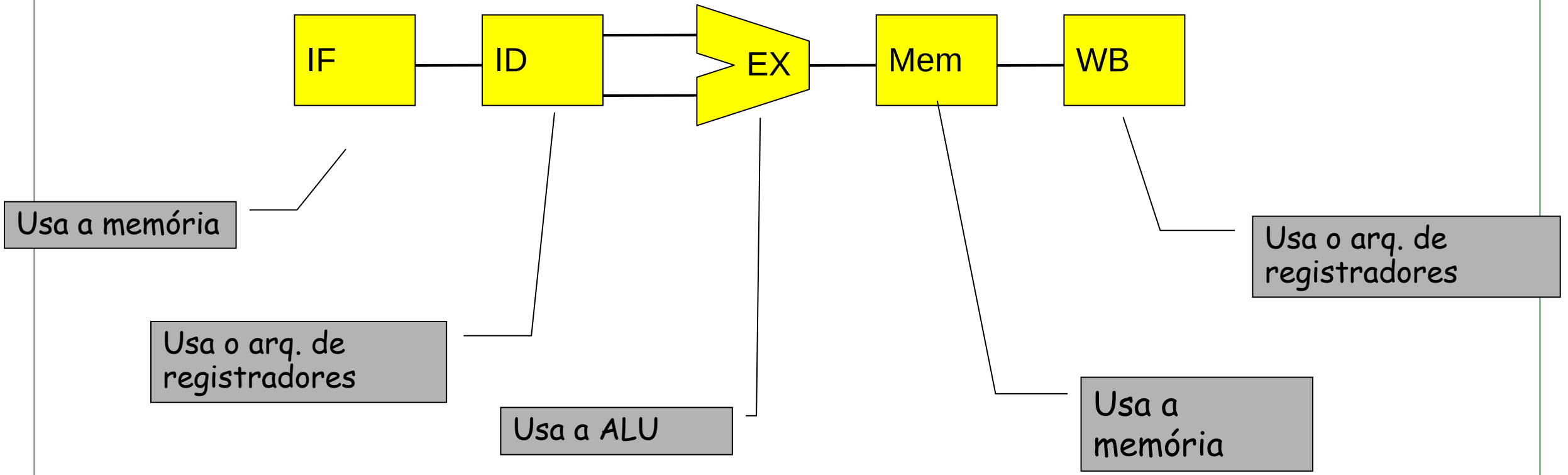
MEM: Memory

- MEM: Se instrução = load:
  - Um dado é lido da memória e armazenado em um registrador temporário
- MEM: Se instrução = store:
  - Um dado é escrito na memória
- MEM: Se instrução = R-type:
  - Armazena o resultado da ALU em um registrador temporário
- Além disso, vários sinais de controle são ativados

WB: Write-back

- WB: Armazena no banco de registradores resultados eventualmente obtidos em estágios anteriores:
  - Dado lido da memória por uma instrução load
  - Resultados da ALU p/ instruções R-type

Pipelining para RISC-V



Pipelining para RISC-V

- A organização de um processador em pipeline não altera o tempo de execução de uma única instrução, mas sim a taxa ou vazão de instruções (throughput) que o processador consegue atingir.

- 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).

- Throughput: trata-se de uma métrica essencial, uma vez que os programas exigem, de uma maneira geral, a execução de milhões a bilhões de instruções.

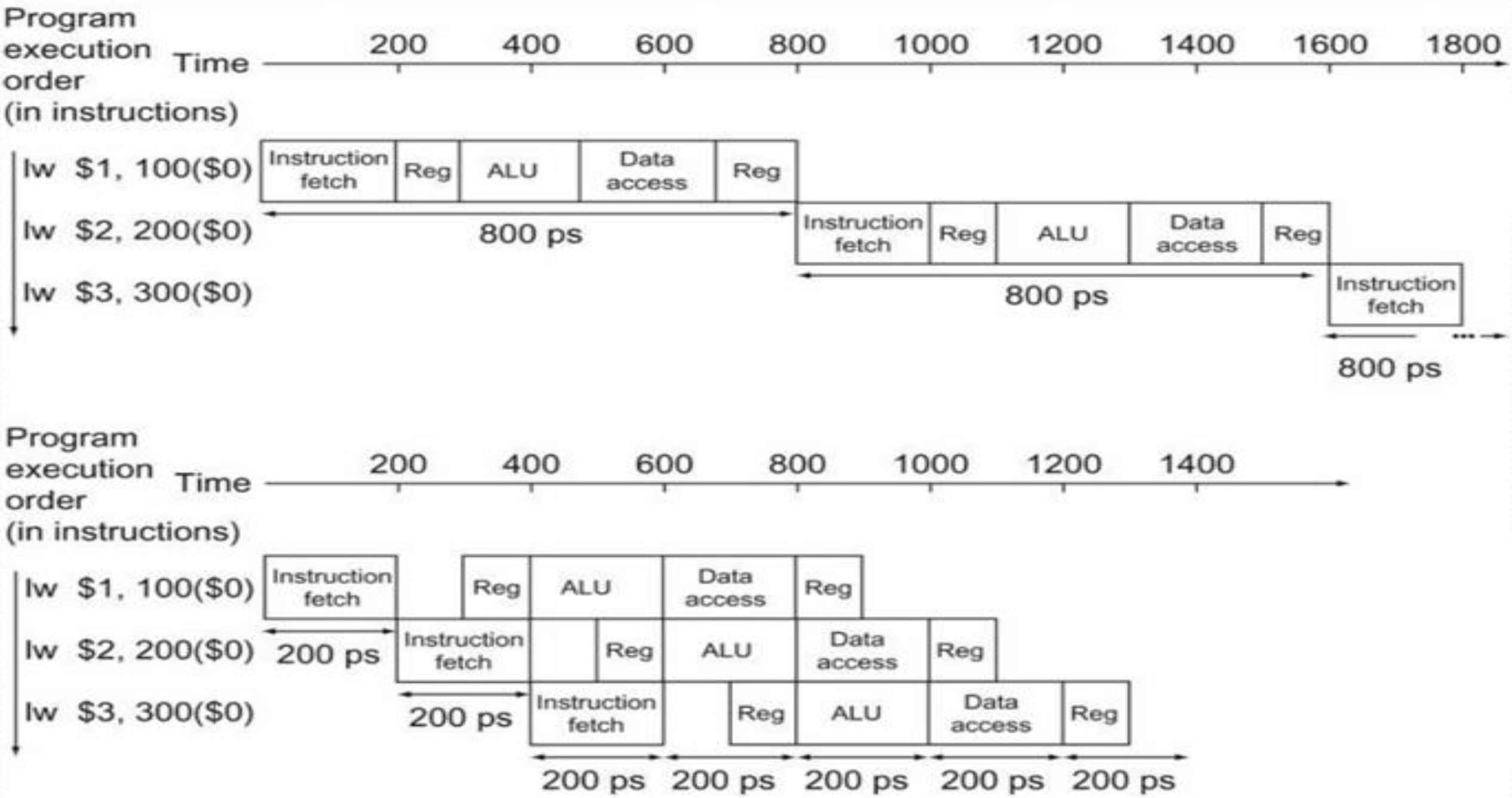
Pipelining para RISC-V

- Monociclo x Pipeline

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (`lw`)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (`sw`)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (`add`, `sub`, `AND`, `OR`, `slt`)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (`beq`)	200 ps	100 ps	200 ps			500 ps

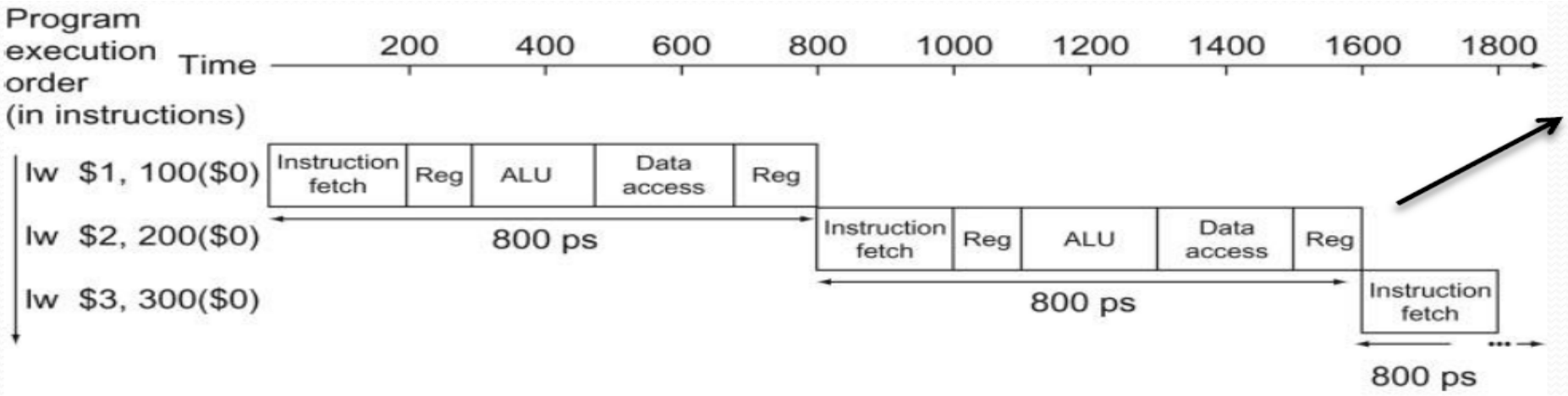
Pipelining para RISC-V

- Monociclo x Pipeline
  - Monociclo: período mínimo de clock = 800ps
  - Pipeline: todos os estágios consomem um ciclo.
    - Logo, o período do clock deve ser longo o suficiente para acomodar a operação mais longa (200ps)

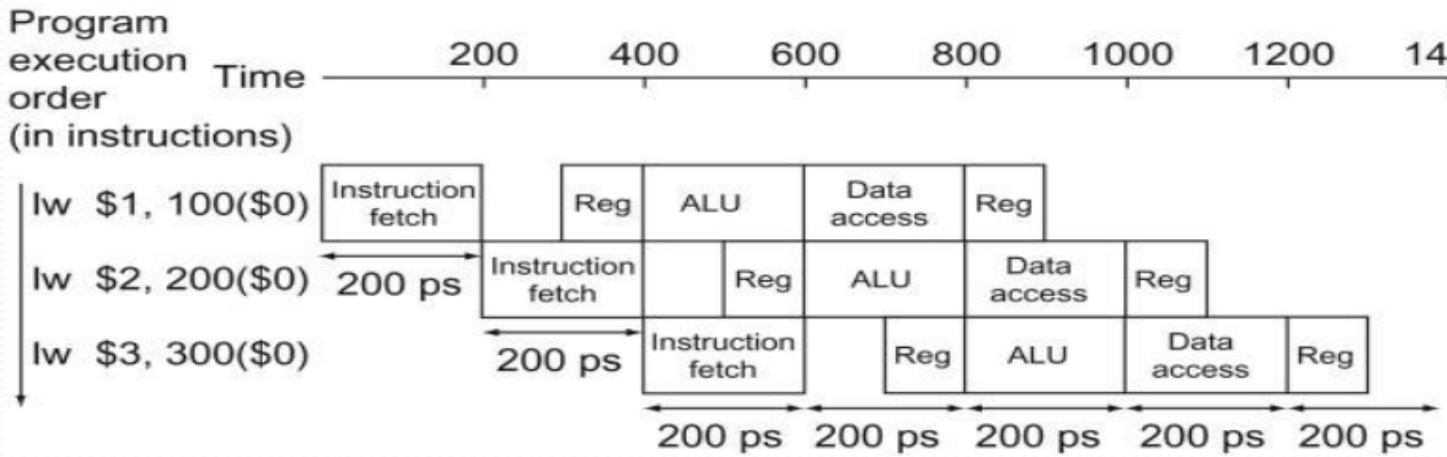

Program execution order (in instructions)
Time
200
400
600
800
1000
1200
1400
1600
1800
lw $1, 100($0)
lw $2, 200($0)
lw $3, 300($0)
Instruction fetch
Reg
ALU
Data access
Reg
800 ps
800 ps
800 ps
Program execution order (in instructions)
Time
200
400
600
800
1000
1200
1400
lw $1, 100($0)
lw $2, 200($0)
lw $3, 300($0)
Instruction fetch
Reg
ALU
Data access
Reg
200 ps
200 ps
200 ps
200 ps
200 ps
200 ps
200 ps

Pipelining para RISC-V

- Monociclo x Pipeline



O tempo entre a primeira instrução *load* e a quarta será de 3 x 800 ps = 2400 ps



Com *pipeline*, a quarta instrução começará a ser executada 600 ps após a primeira.

Arquitetura tipo Pipelined

- O conjunto de instruções RISC-V foi planejado visando uma implementação "eficiente" com pipeline.
  - Todas as instruções possuem o mesmo tamanho (32 bits):
    - Facilita a busca e a decodificação, que podem ser feitas em um ciclo cada (no 1º e 2º estágios da pipeline, respectivamente).
  - Os campos dos registradores que fornecem operandos estão localizados no mesmo lugar em cada instrução.
    - Esta simetria abre a possibilidade de o segundo estágio da pipeline iniciar a leitura do arquivo de registrador ao mesmo tempo em que o hardware está identificando a instrução recebida.

Arquitetura tipo Pipelined

- O conjunto de instruções RISC-V foi planejado visando uma implementação "eficiente" com pipeline.
  - Operandos em memória somente aparecem nas instruções load e store:
    - Esta restrição de projeto possibilita que o 3º estágio (execução) seja utilizado para o cálculo do endereço de memória e a referida posição seja acessada no estágio seguinte.
    - Se fosse permitida a especificação de operandos em memória, como acontece na arquitetura x86, os estágios 3 e 4 seriam expandidos em: (1) estágio de endereço, (2) estágio de acesso à memória e (3) estágio de execução.

Arquitetura tipo Pipelined

- O conjunto de instruções RISC-V foi planejado visando uma implementação "eficiente" com pipeline.

  - Alinhamento dos operandos em memória:

    - No RISC-V, as palavras sempre iniciam em endereços de memória que são múltiplos de 4.

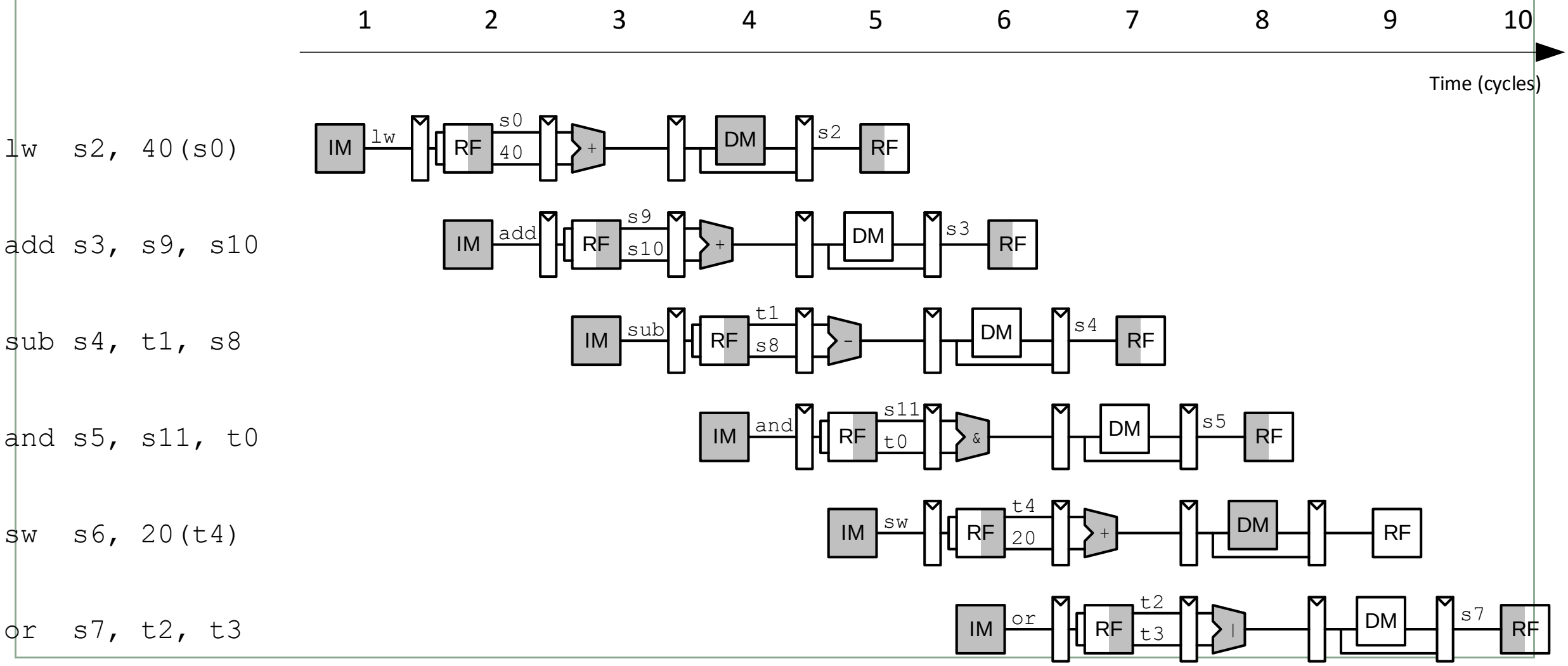
    - Por causa desta restrição (ou garantia), um acesso à memória pode ser feito em apenas um estágio.

Pipelining para RISC-V - Exemplo

- Dado os 5 estágios anteriores, em regime:
  - uma nova instrução é inicia a cada ciclo
  - uma instrução termina a execução a cada ciclo
  - logo, a produção (throughput) é proporcional ao estágio mais lento.
- Infelizmente existem fatores que dificultam atingir o desempenho máximo (hazards)

	clock cycle								
Instruction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
i	IF	ID	EX	MEM	WB				
i+1		IF	ID	EX	MEM	WB			
i+2			IF	ID	EX	MEM	WB		
i+3				IF	ID	EX	MEM	WB	
i+4					IF	ID	EX	MEM	WB

Pipelining para RISC-V - Exemplo

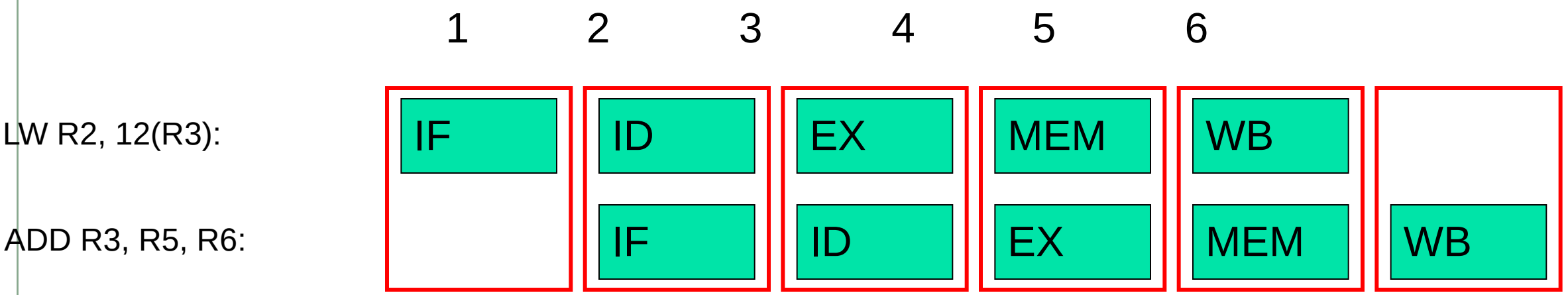


Exemplo

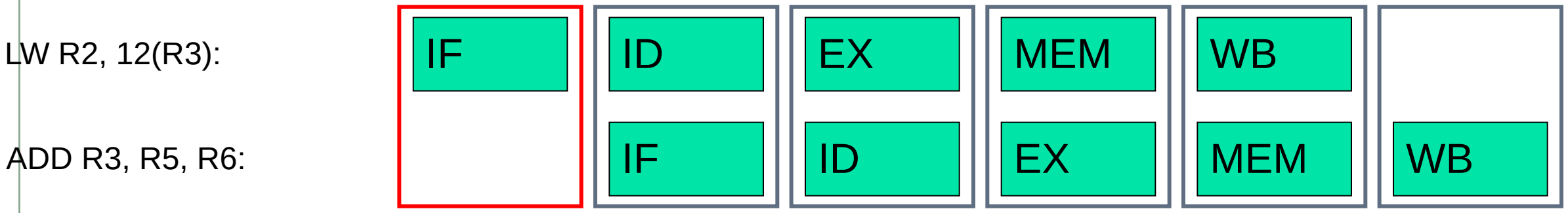
- Considere o seguite código (instruções I1 e I2)

```
I1:  LW R2, 12(R3)
I2:  ADD R4, R5, R6
```

- Execução total= 6 estágios



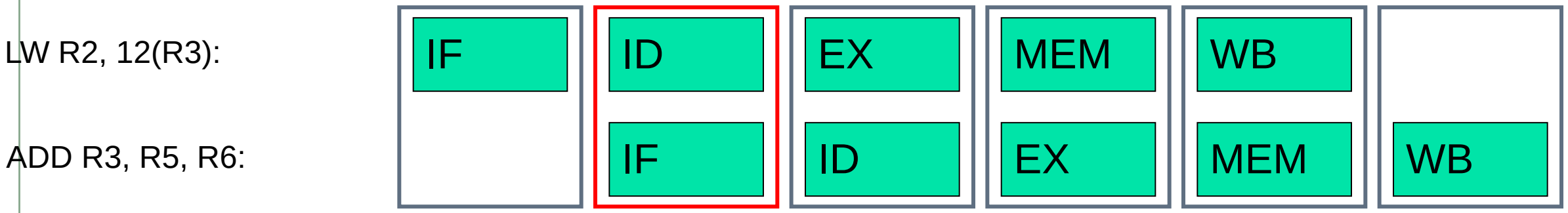
Exemplo



- Passo 1: I1 em IF, I2 não emitida
  - Envia o PC p/ memória para buscar I1
    - Obtém I1 um ciclo depois
  - PC= PC+4 (shift-left 2)
    - Atualizado no mesmo ciclo

Issued= emitada (enviada para execução)

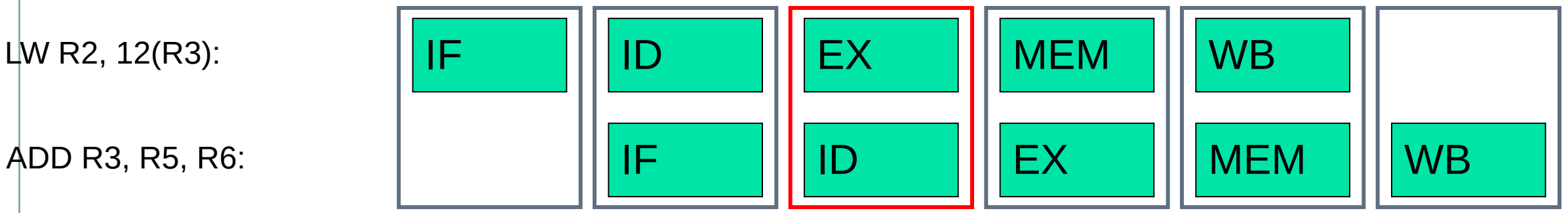
Exemplo



- Segundo passo: I1 em ID, I2 em IF
  - I1:

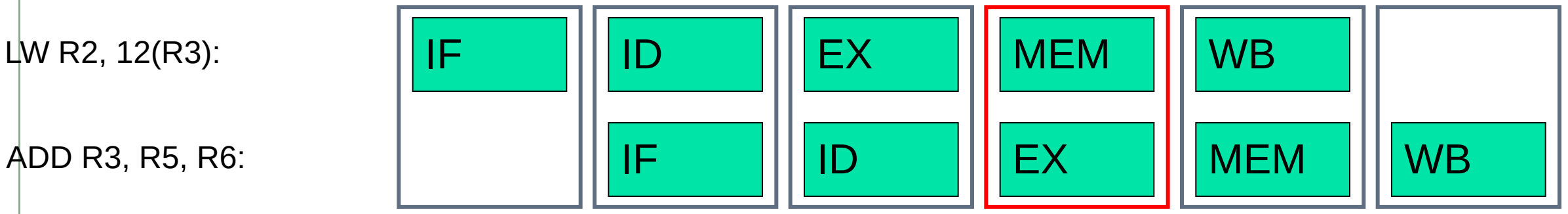
    Decodifica a instrução
    - Lê o conteúdo do registrador R3
    - Obtém a constante 12 (imediato), e faz a extensão de sinal
  - I2:
    - Envia o PC p/ memória para buscar I2
    - PC= PC+4

Exemplo



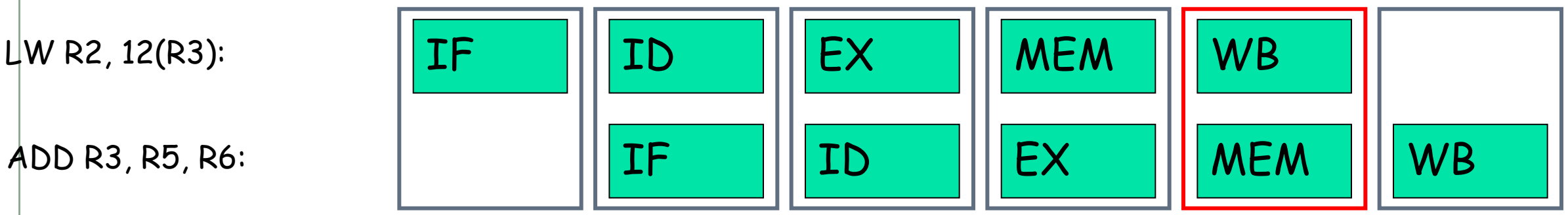
- Terceiro Passo: I1 em EX, I2 em ID
  - I1:
    - Soma o conteúdo de R3 c/ a constante 12, calculando endereço de acesso à memória.
  - I2:
    - Decodifica a instrução
      - Lê o conteúdo dos registradores R5 e R6

Exemplo



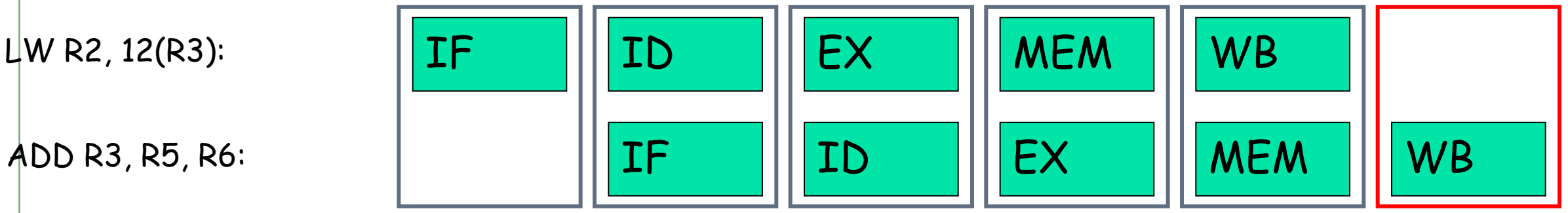
- Quarto Passo: I1 em MEM, I2 em EX
  - I1:
    - Envia o endereço calculado em EX p/ a Memória
    - Lê a double word da memória
  - I2:
    - Soma o conteúdo de R5 c/ R6

Exemplo



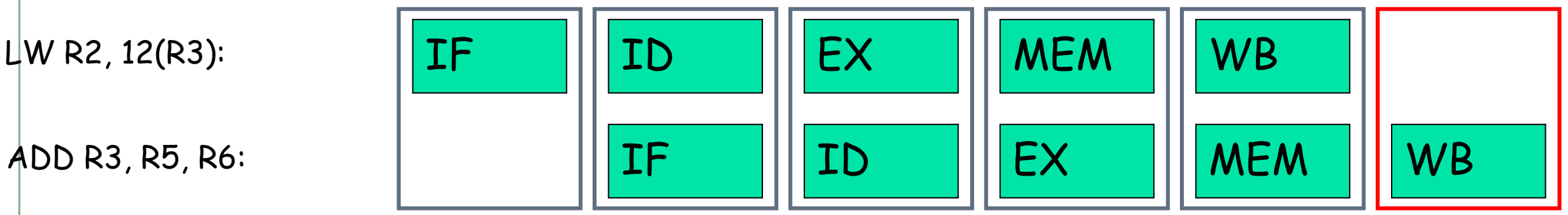
- Quinto Passo: I1 em WB, I2 em MEM
  - I1:
    - Grava o dado lido da memória no registrador R2
  - I2:
    - Nada a fazer (instrução add não acessa a memória)

Exemplo



- Sexto Passo: I1 terminada, I2 in WB
  - I2:
    - Grava o valor calculado em EX no registrador R3

Exemplo

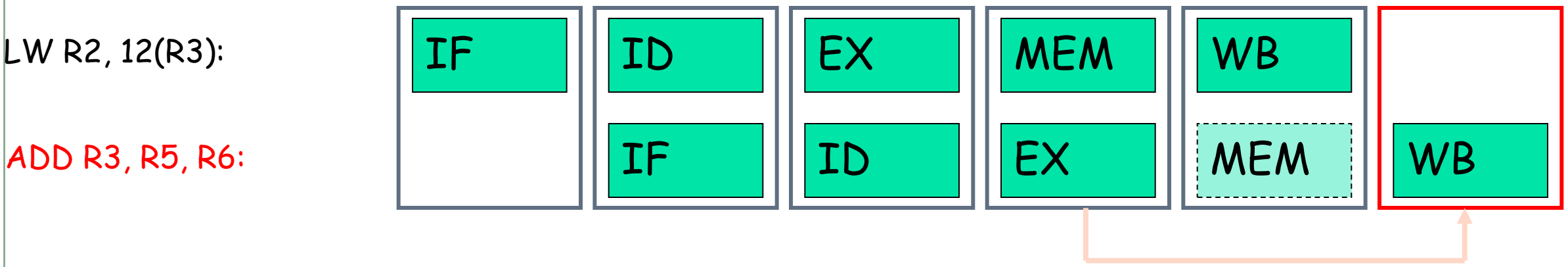


- Sexto Passo: I1 terminada, I2 in WB
  - I2:
    - Grava o valor calculado em EX no registrador R3

- Fim da execução: Duas instruções foram executadas em 6 ciclos de clock
  - CPI médio: 3 ciclos de clock
  - Duração de cada instrução: 5 ciclos de clock

Pipelining e Registradores

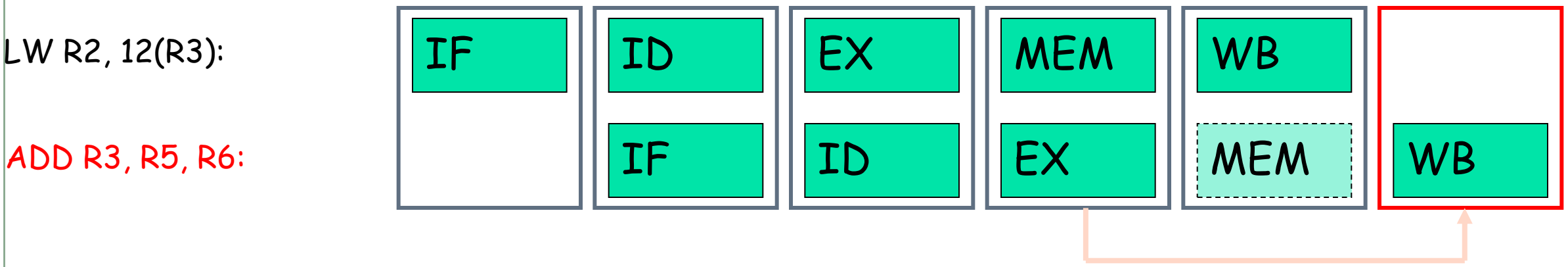
- Às vezes um valor é calculado em um estágio, mas usado em um outro estágio não adjacente.



- Pergunta: como esse valor é comunicado do estágio EX para o estágio WB ?

Pipelining e Registradores

- Às vezes um valor é calculado em um estágio, mas usado em um outro estágio não adjacente.

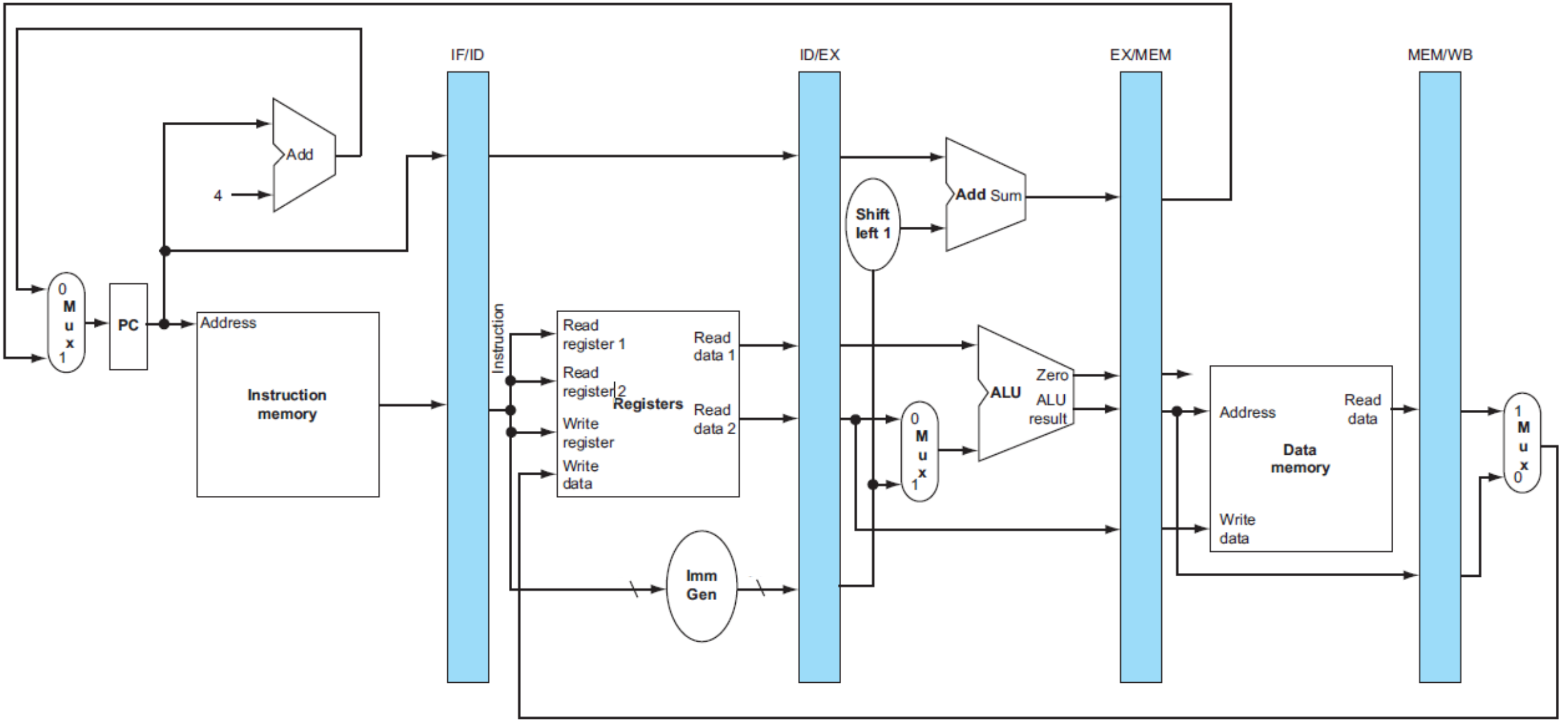


- Pergunta: como esse valor é comunicado do estágio EX para o estágio WB ?
- Resposta: Usando registradores de pipeline

Registradores de Pipelining

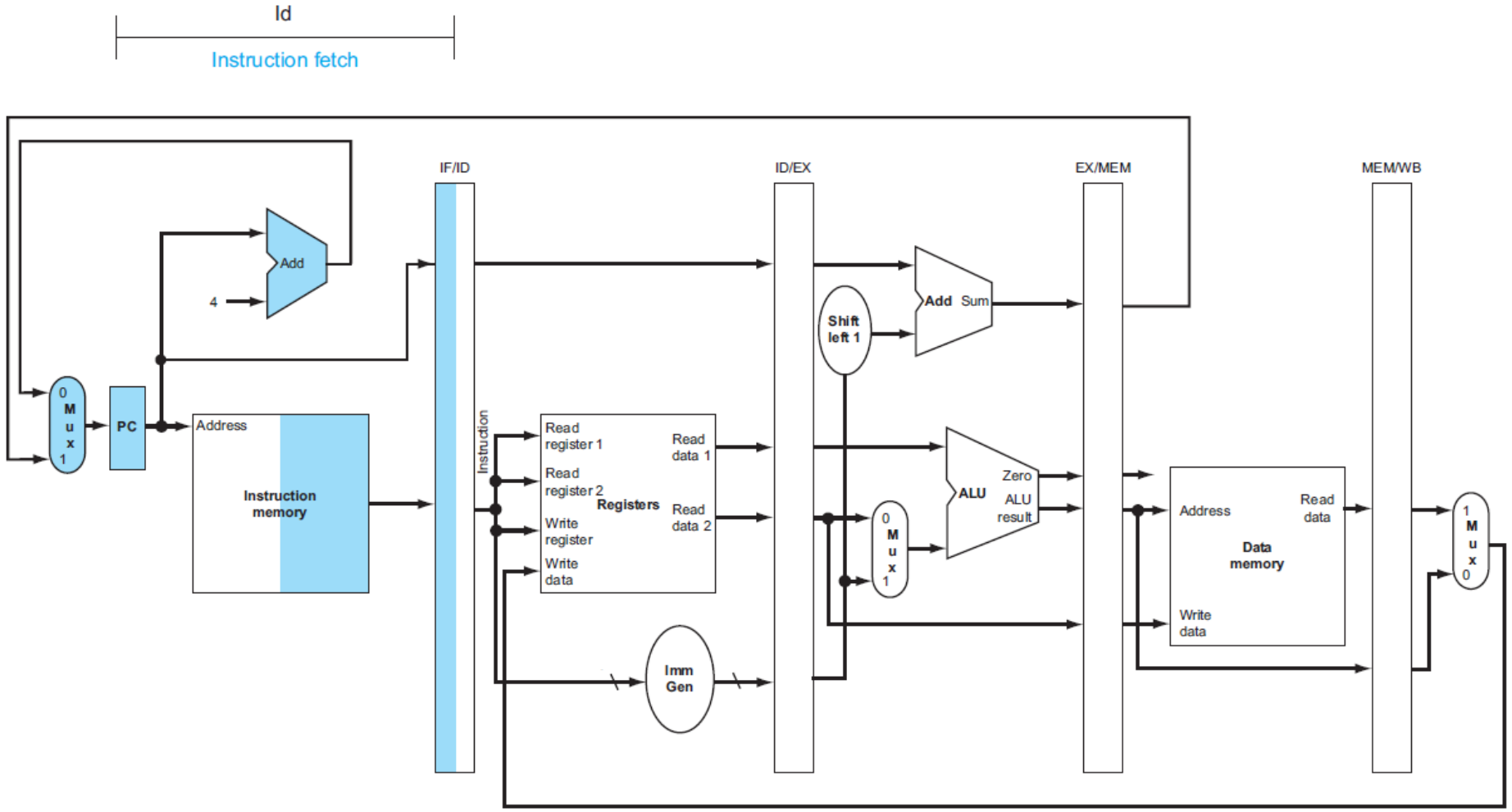
- A técnica descrita anteriormente necessita de registradores de pipelining
- Para o datapath RISC-V descrito, temos 4 grupos de registradores de pipelining.
- Cada grupo é usado para armazenar a saída de um estágio, e passá-la à entrada do estágio seguinte.
- Podemos identificar esses grupos de acordo c/ os estágios adjacentes:
  - Grupo 1: IF/ID
  - Grupo 2: ID/EX
  - Grupo 3: EX/MEM
  - Grupo 4: MEM/WB
- Dessa forma, toda informação obtida ou computada é guardada para ser usada no contexto da instrução em questão.

Registradores de Pipelining



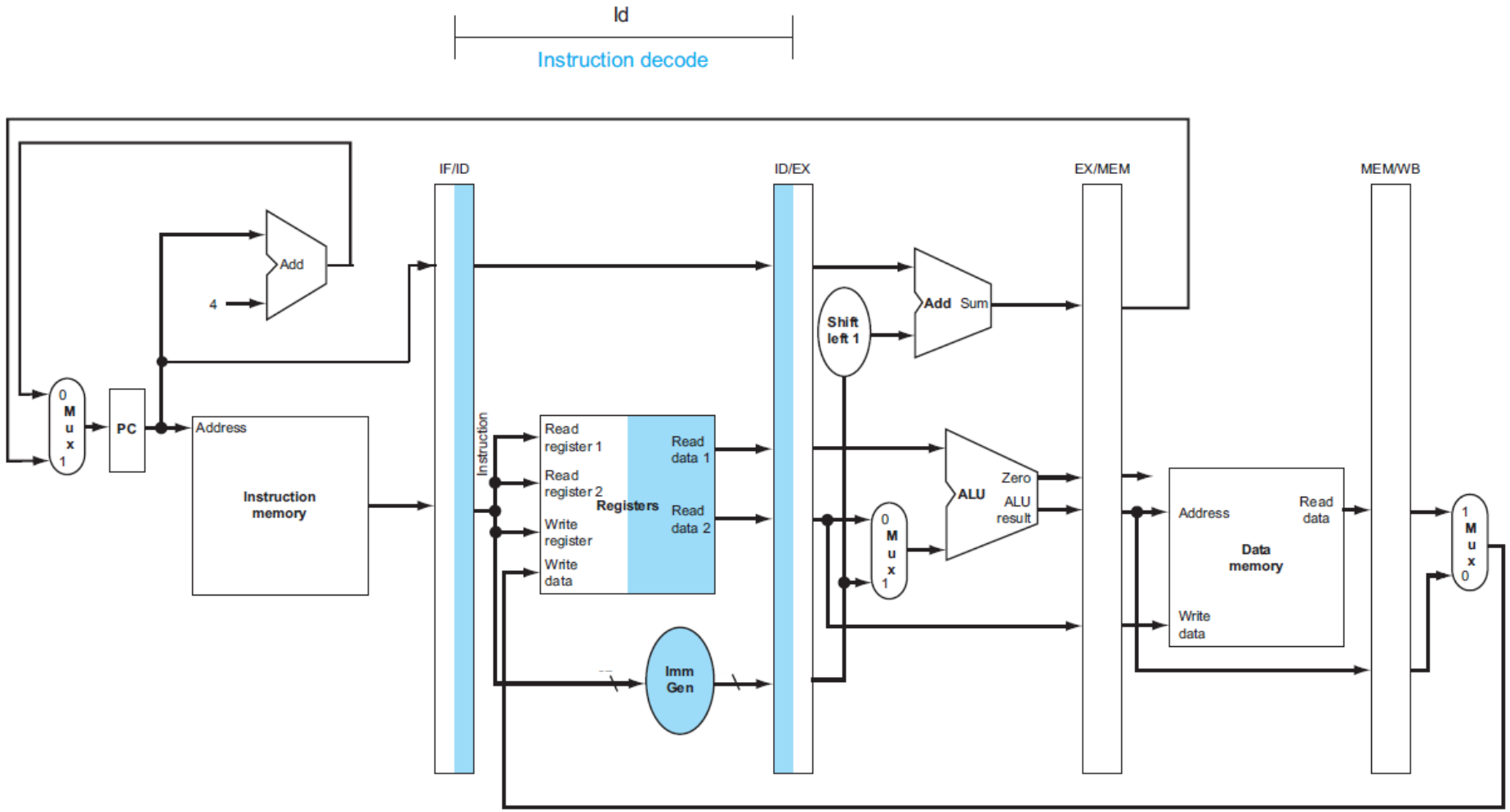
Pipelining para RISC-V

- Executando lw



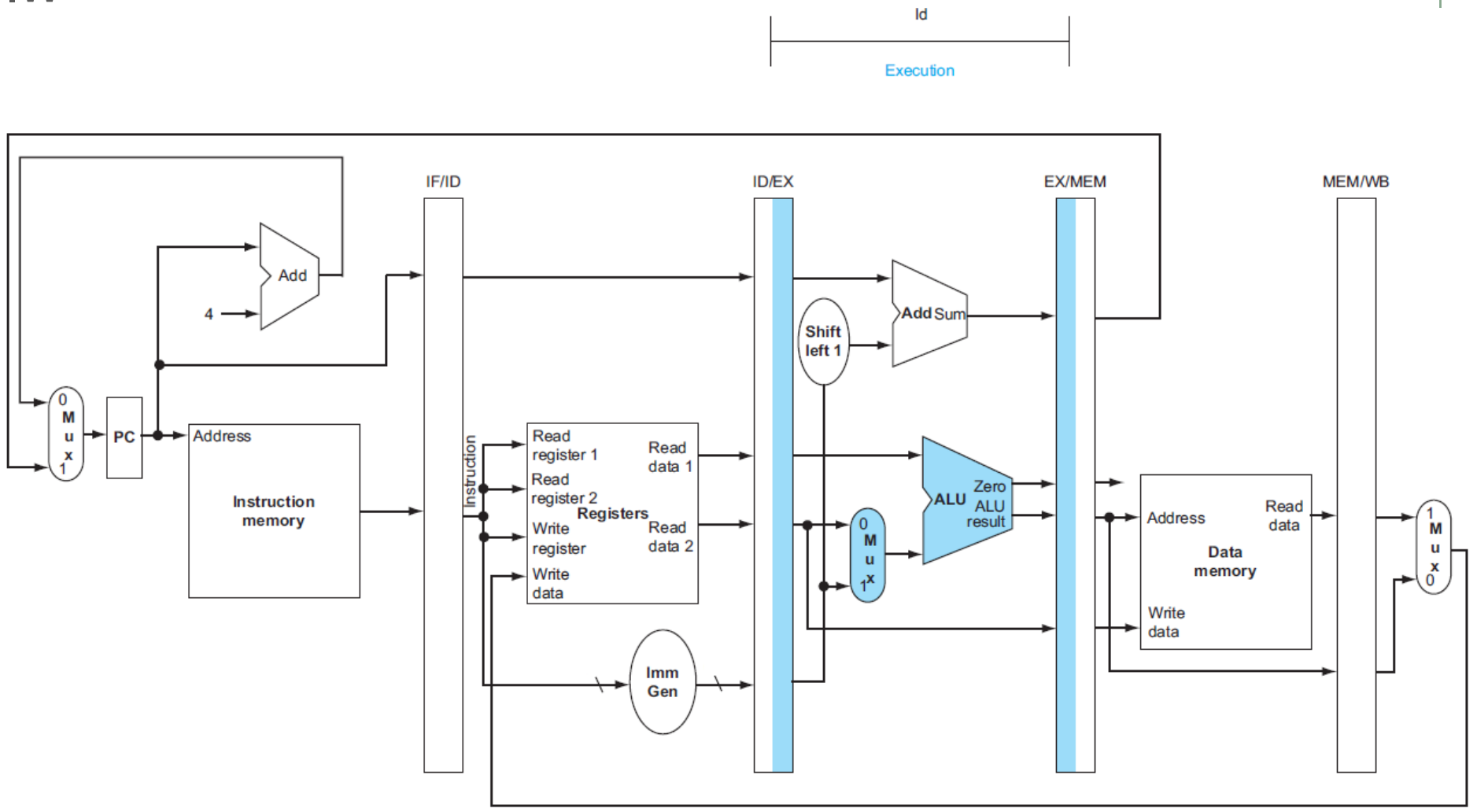
Pipelining para RISC-V

- Executando lw



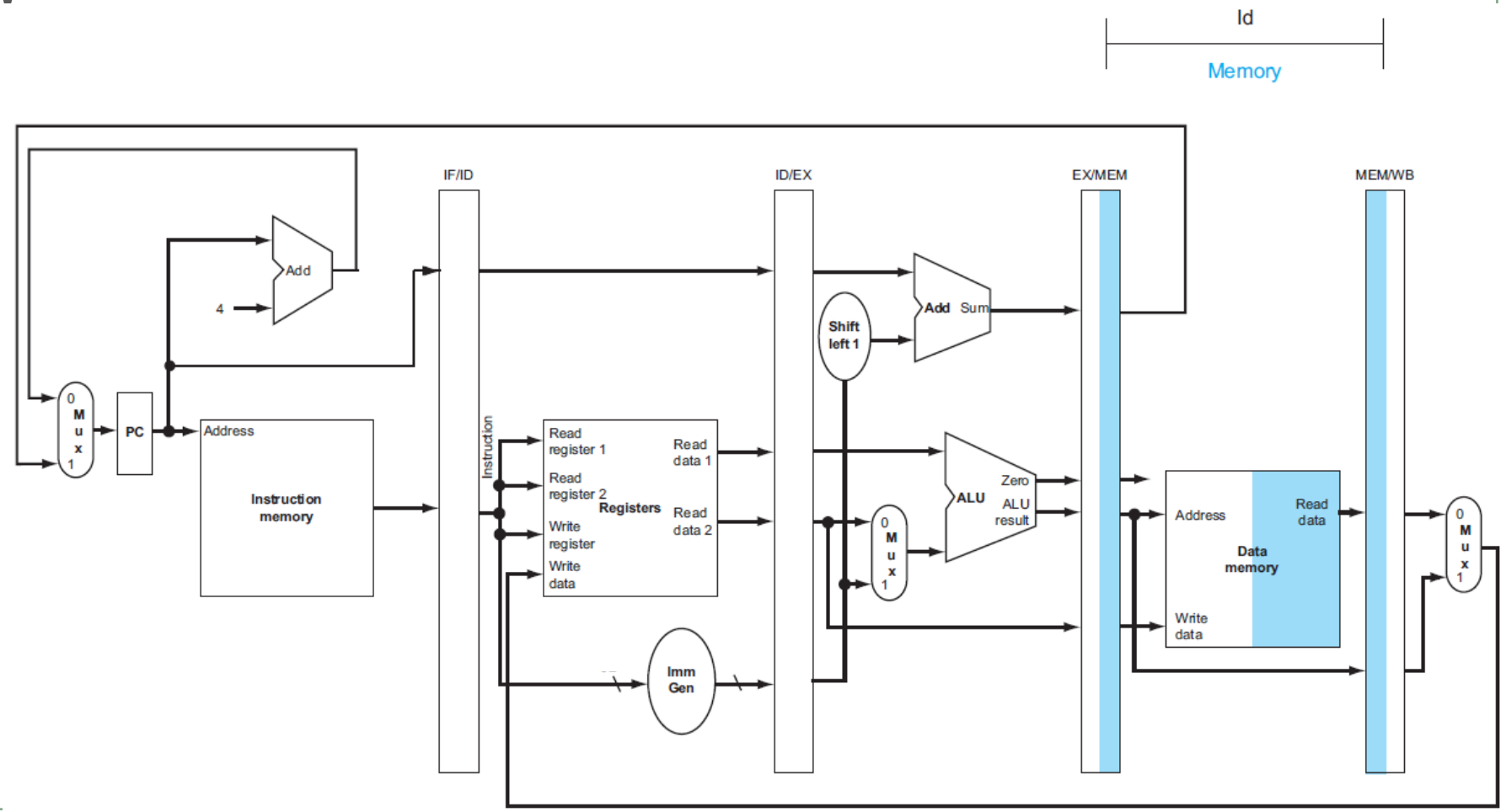
Pipelining para RISC-V

- Executando lw



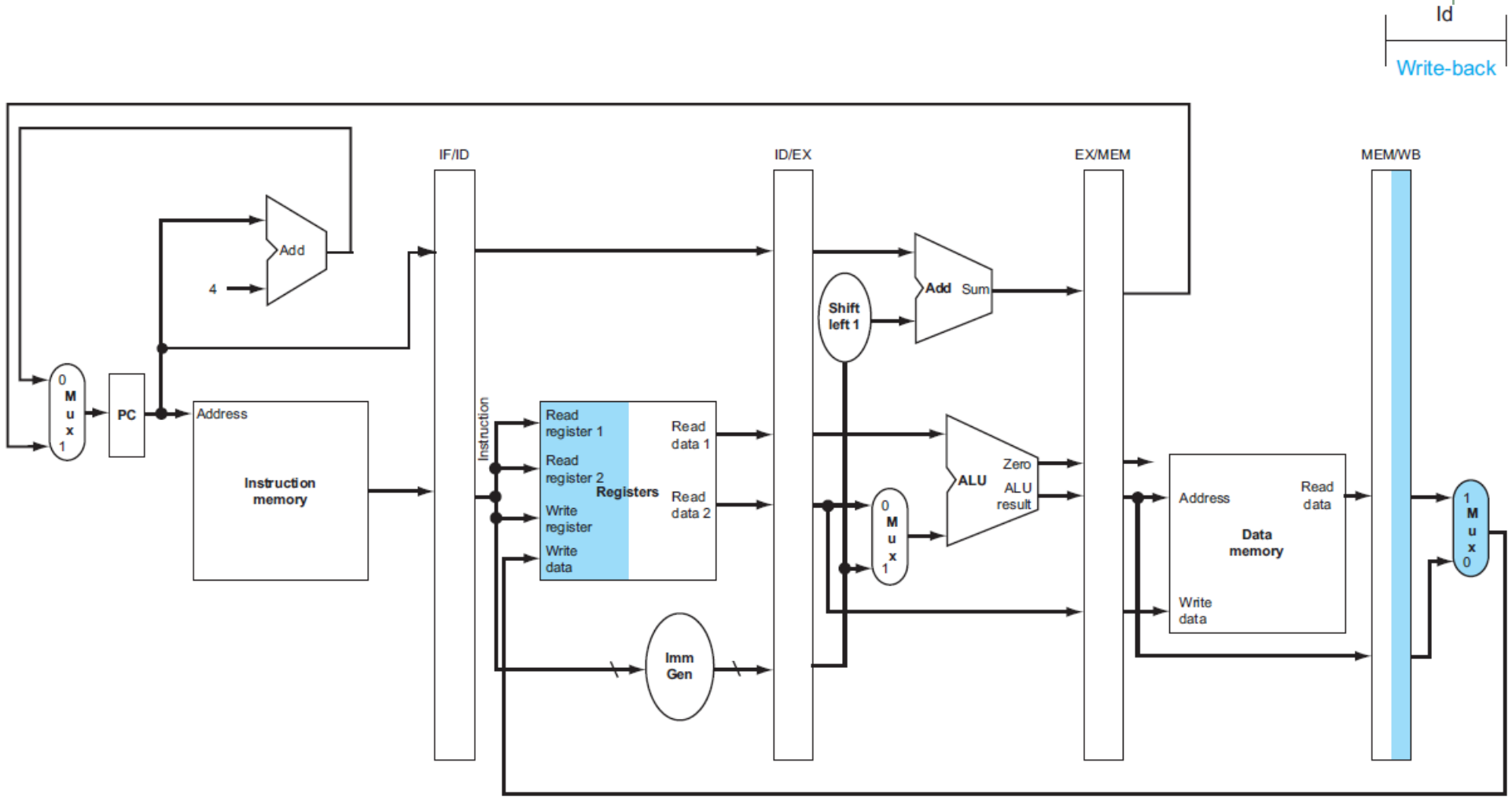
Pipelining para RISC-V

- Executando lw



Pipelining para RISC-V

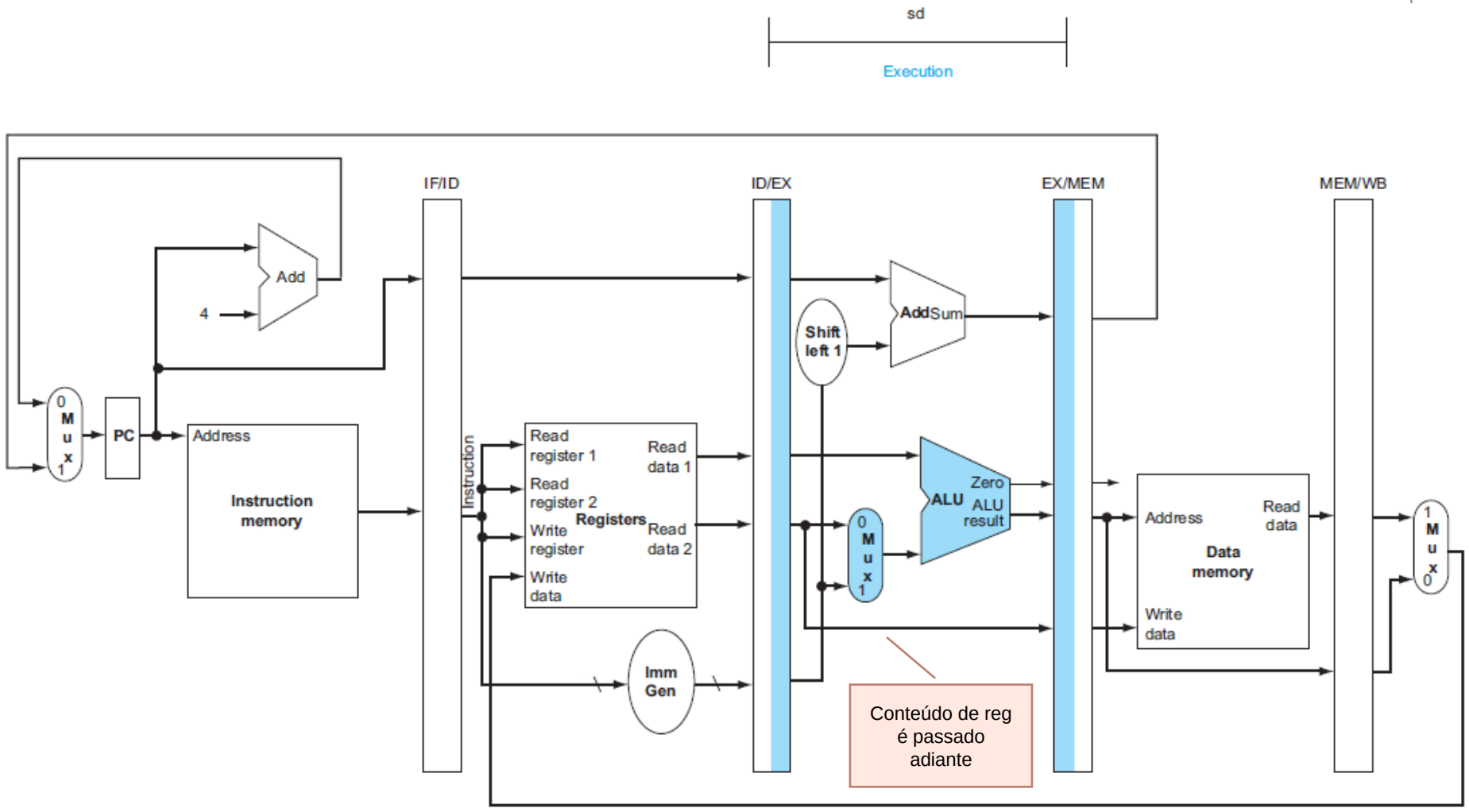
- Executando lw



Pipelining para RISC-V

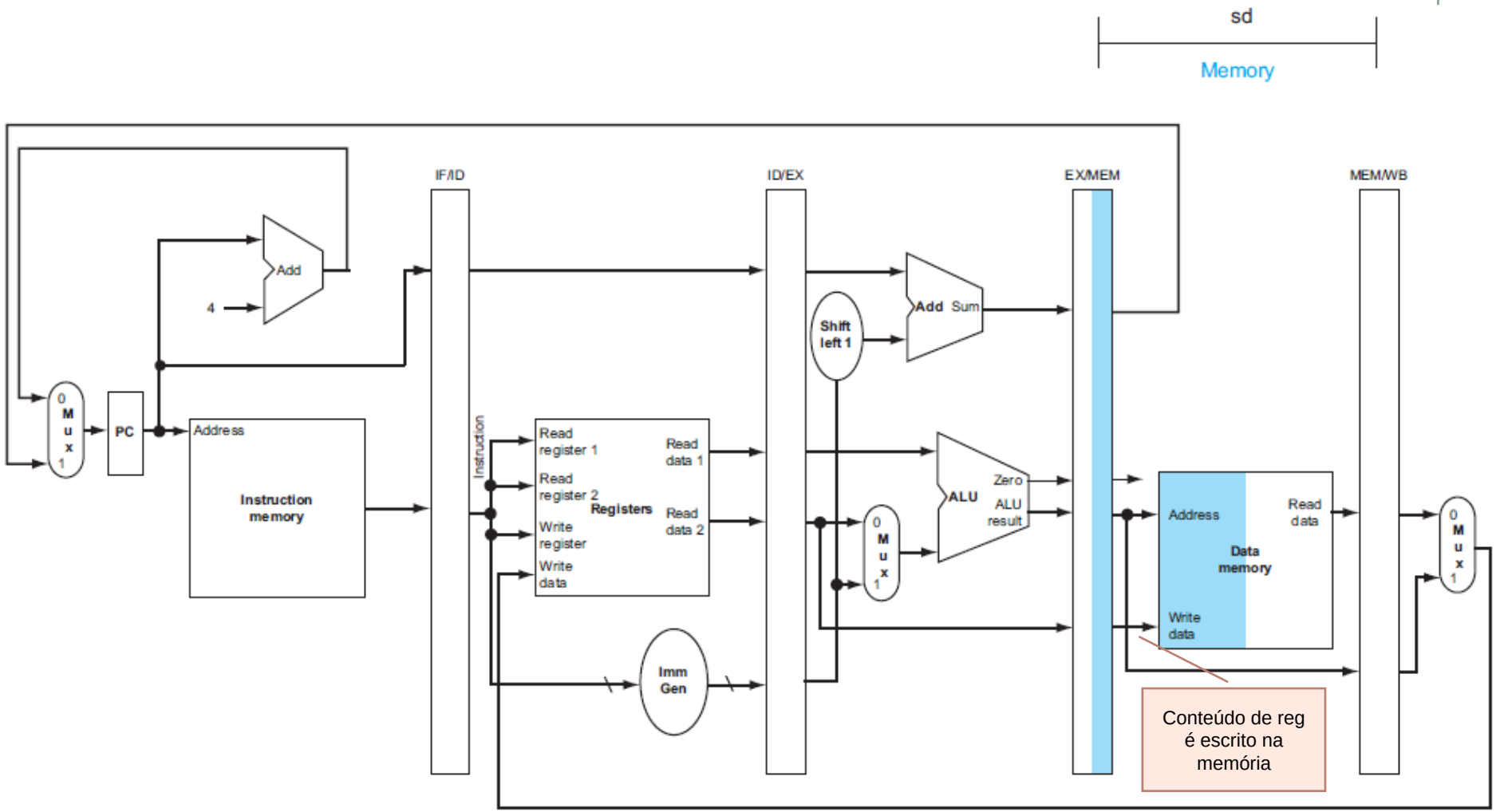
- Executando sw

Estágios 1 e 2 idêntico aos de SW



Pipelining para RISC-V

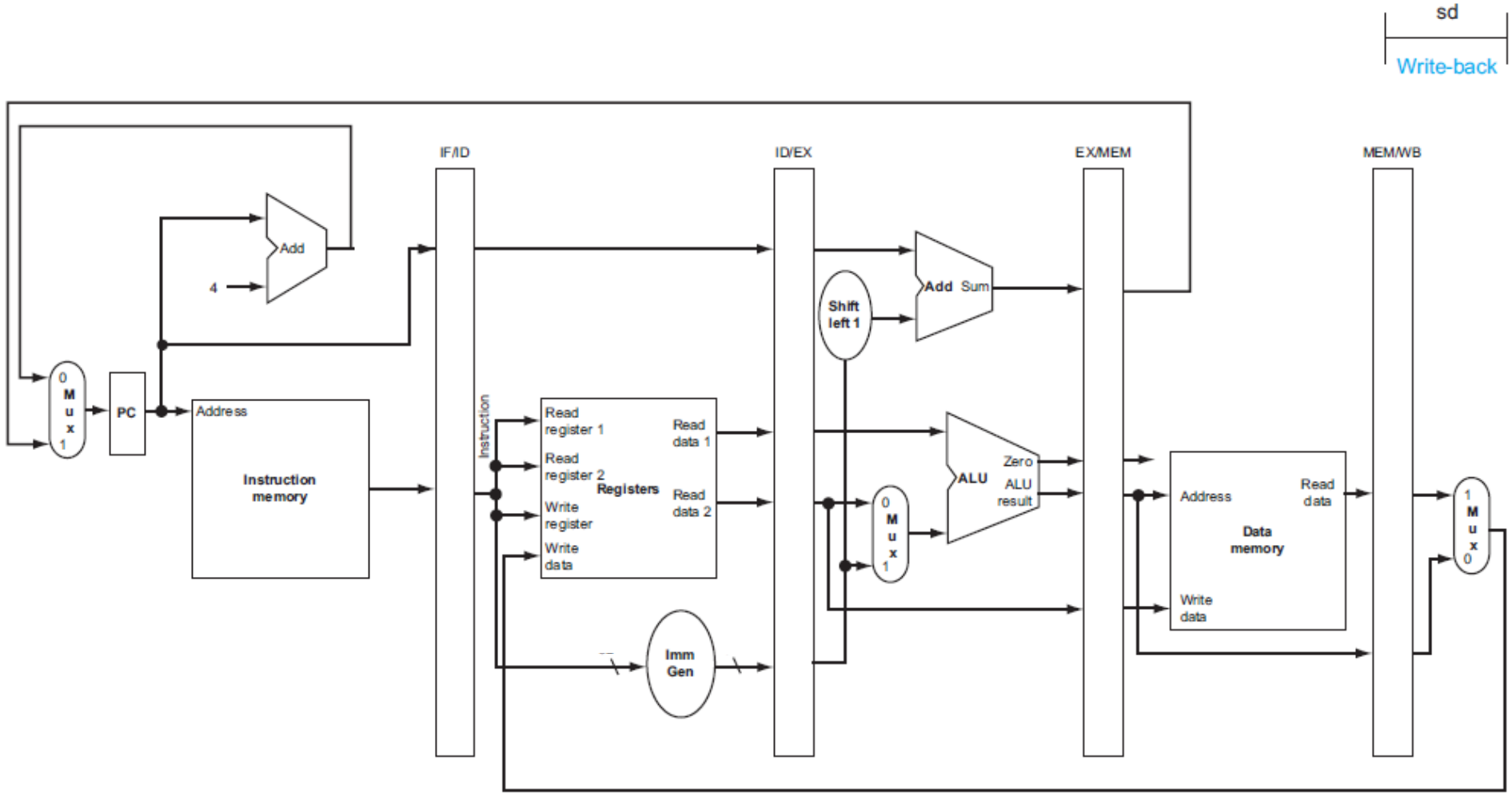
- Executando sw



Pipelining para RISC-V

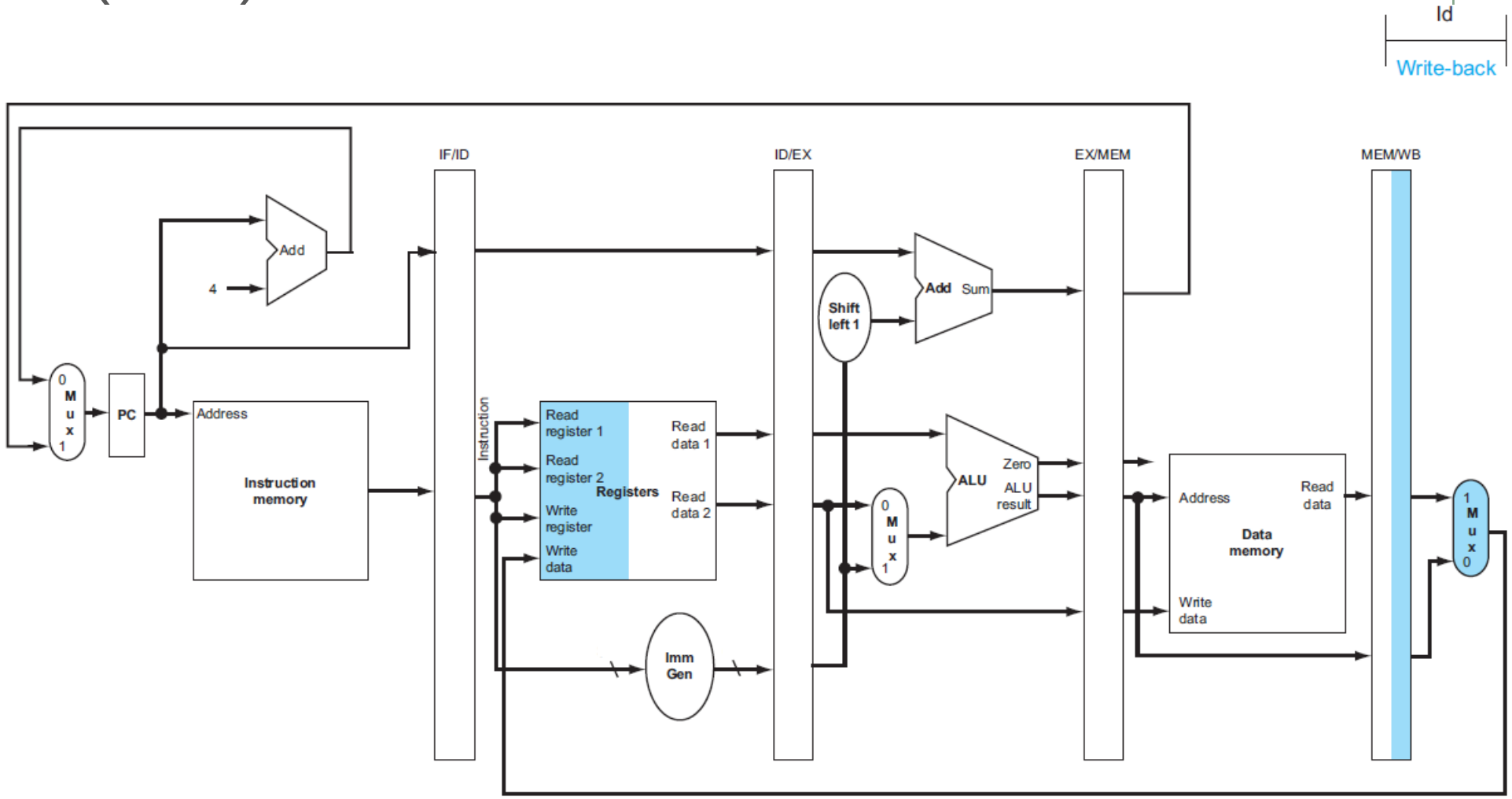
- Executando sw

Nada acontece no 5º estágio



Pipelining para RISC-V

- Executando lw (BUG)

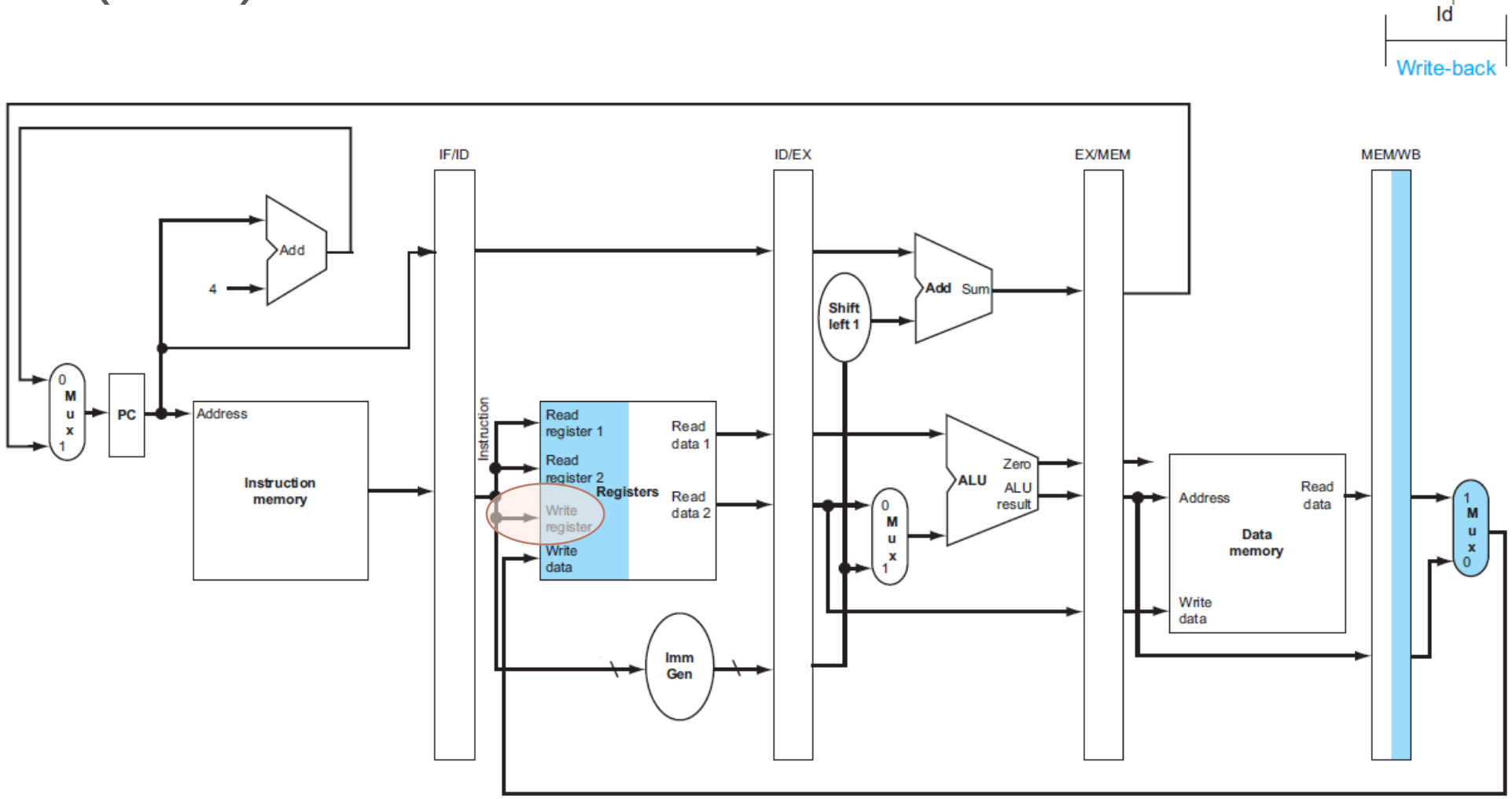


Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

Pipelining para RISC-V

- Executando lw (BUG)

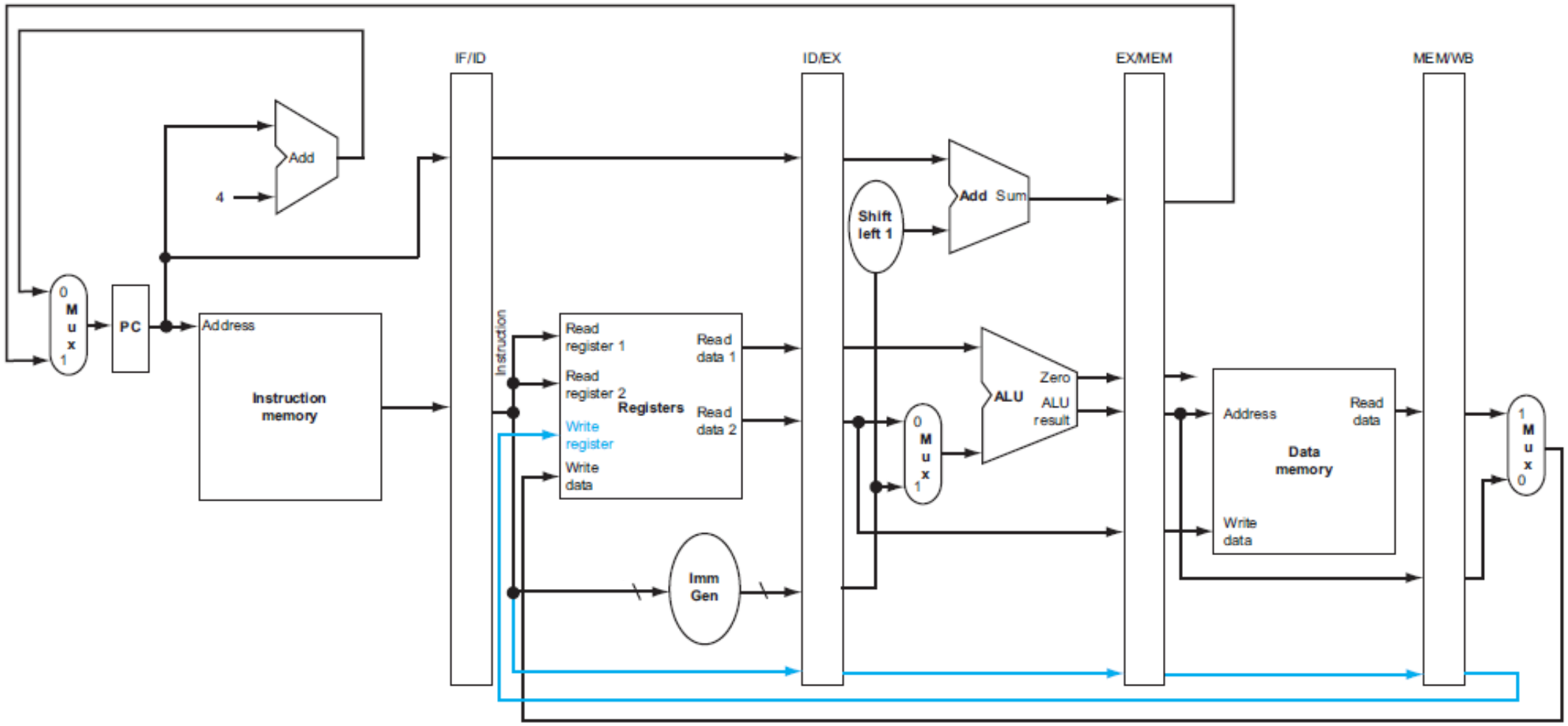
O endereço do registrador a ser escrito deve viajar junto com a instrução lw



Fonte: Prof. José Luís Güntzel - Arquitetura e Organização de Computadores II

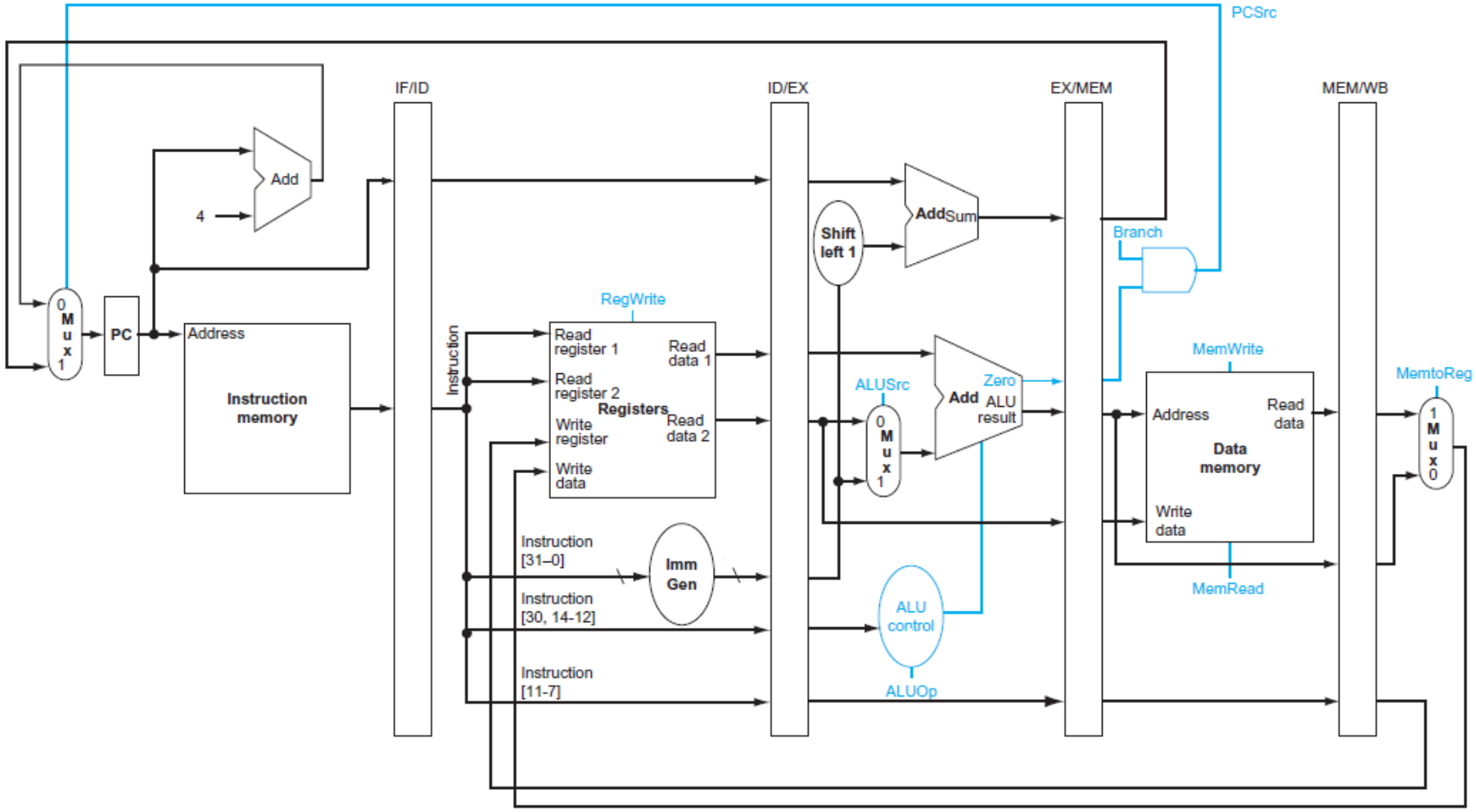
Pipelining para RISC-V

- Executando lw (Corrigido)



Pipelining para RISC-V

- Sinais de Controle



Pipelining para RISC-V

- Unidade de Controle
  - IF: os sinais de controle para ler a instrução e para escrever em PC estão sempre ativos.
  - ID: não há linhas de controle opcionais, pois a operação neste estágio sempre é a mesma, independentemente da instrução.
  - EX: os sinais que precisam ser ajustados são RegDst, ALUOp e ALUSrc.
  - MEM: as linhas de controle deste estágio são Branch, MemRead e MemWrite.
  - WB: as linhas de controle deste estágio são MemtoReg, que decide entre enviar o resultado da ALU ou um valor de memória para o arquivo de registradores, e RegWrite, que escreve o valor selecionado.

Pipelining para RISC-V

- Unidade de Controle
  - É possível aproveitar ao máximo os sinais de controle do RISC-V monociclo, e a mesma lógica de controle para:
    - A ULA
    - O desvio condicional
    - O multiplexador que controla a fonte do dado do registrador destino
    - E demais linhas apresentadas anteriormente
  - Os sinais de controle são essencialmente os mesmos do RISC-V monociclo
  - A única particularidade é que eles precisam “viajar” pelos estágios juntamente com a instrução

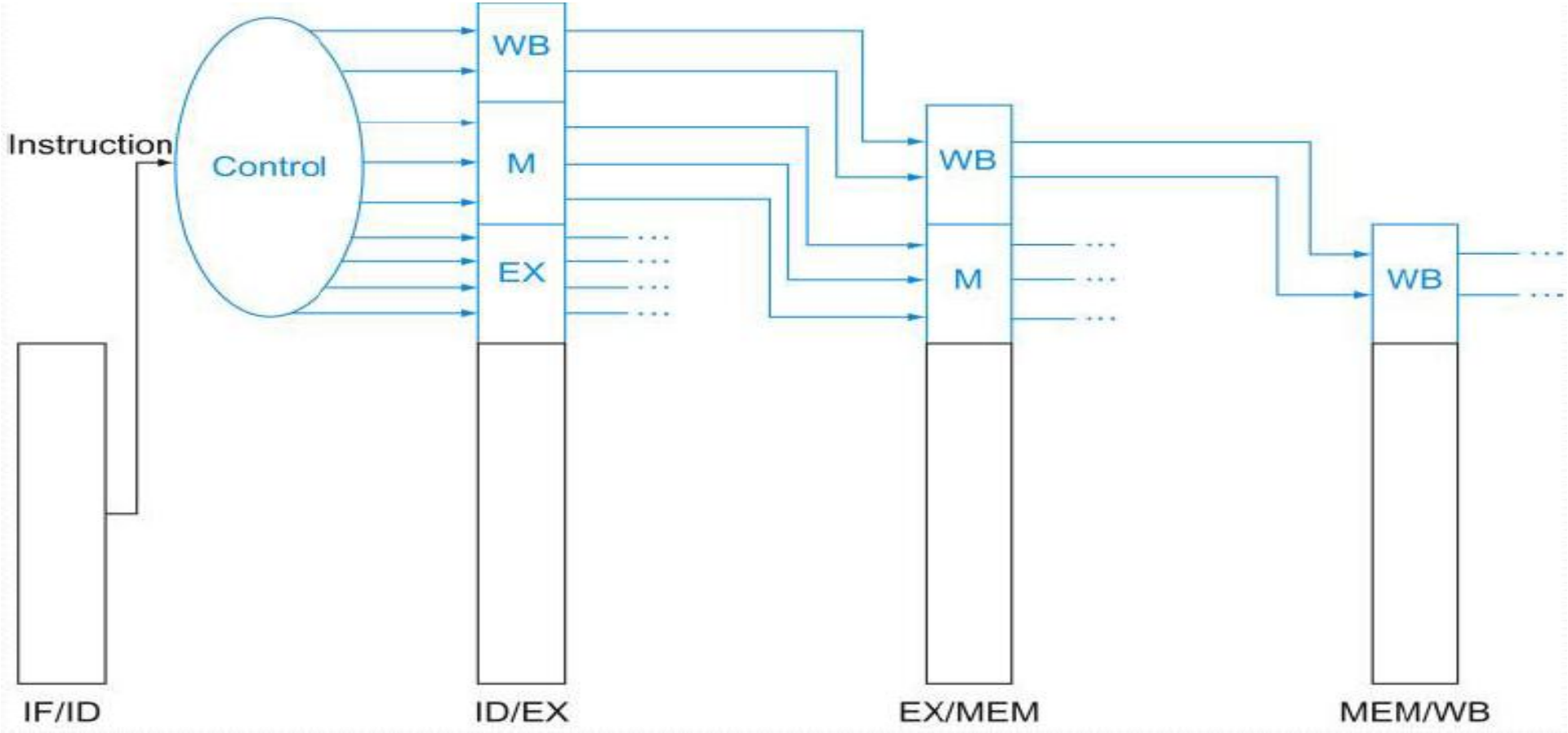
Pipelining para RISC-V

- Unidade de Controle

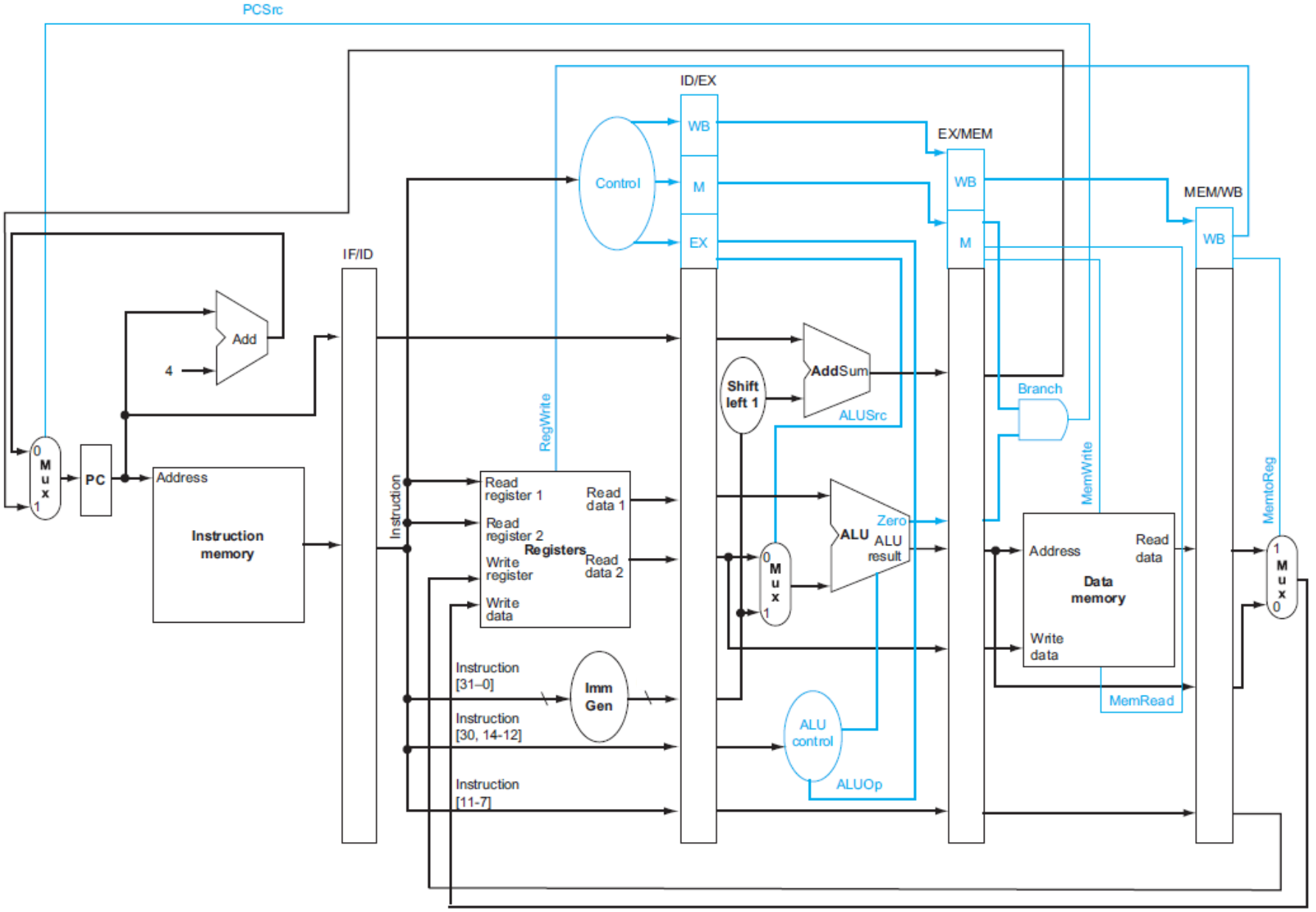
Instruction	Execution/address calculation stage control lines		Memory access stage control lines			Write-back stage control lines	
	ALUOp	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	10	0	0	0	0	1	0
ld	00	1	0	1	0	1	1
sd	00	1	0	0	1	0	X
beq	01	0	1	0	0	0	X

Pipelining para RISC-V

- Sinais de Controle



- Sinais de Controle



Conclusão

- Pipelining: forma efetiva de se melhorar o desempenho de microprocessadores.
  - Diminui o CPI
  - Diminui o tempo de um ciclo de clock
- Arquiteturas RISC: facilitam a implementação de pipelining;
- Uso generalizado em toda arquitetura projetada nos últimos 20 anos;
- Problemas associados com pipelining
  - Structural Hazards
  - Data Hazards
  - Controle mais complexo