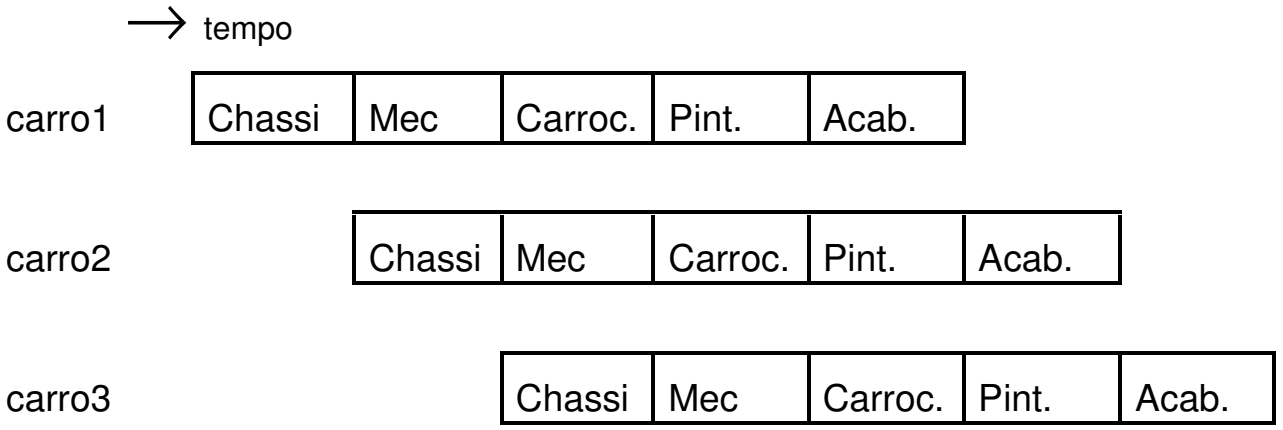
Capítulo 6
Pipeline

Pipeline: analogia com linha de produção

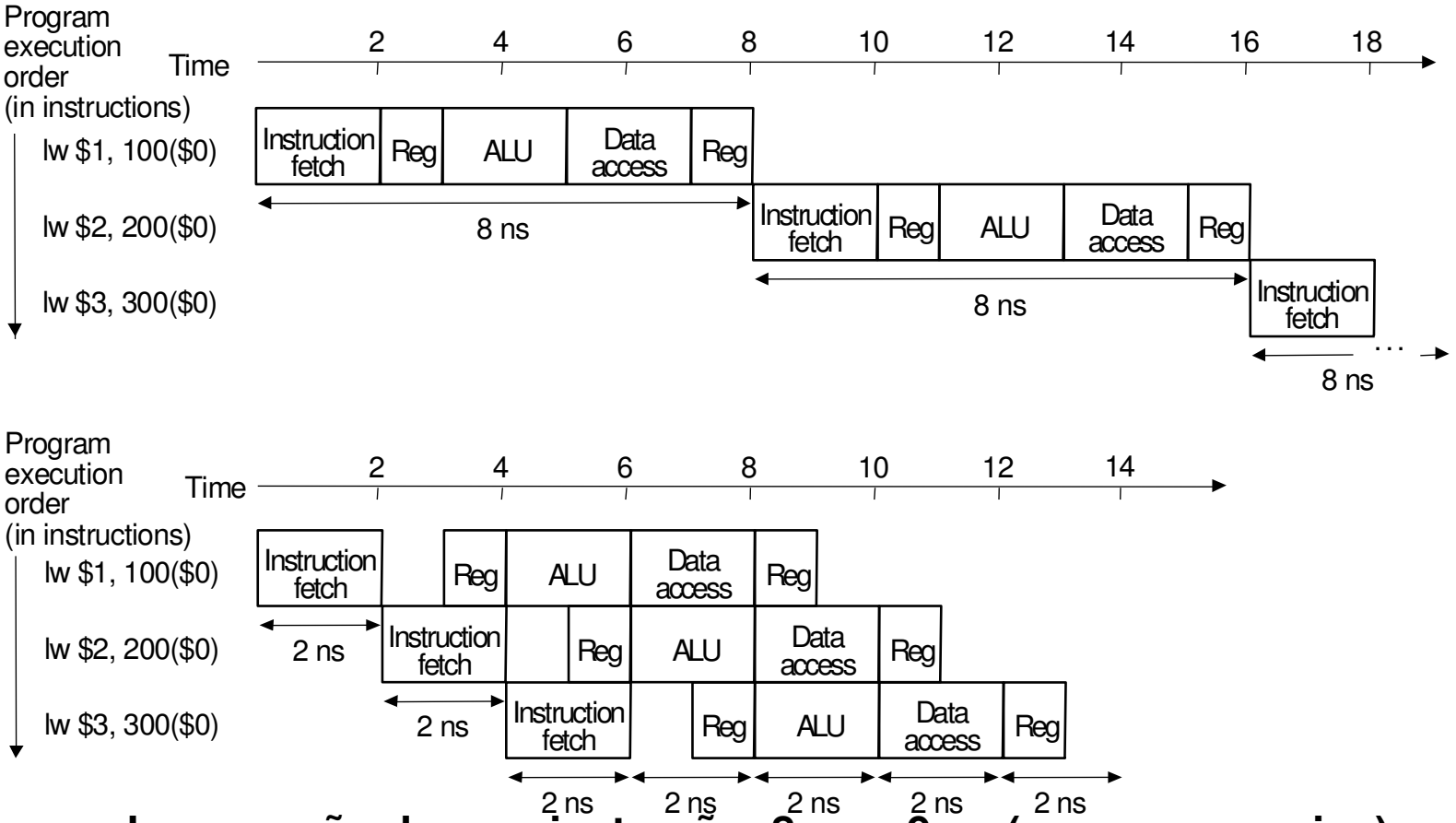


- **Tempo de fabricação de um carro: C+M+C+P+A**
- **Taxa de produção (carros/h): medido na saída**
  - ***Throughput* (vazão): depende do número de estágios e do balanceamento**

- **Objetivo do pipeline:**
  - **distribuir e balancear o tempo em cada estágio**
  - **otimizar o uso de HW dos estágios (taxa de ocupação)**
  - **resumo: aumentar a velocidade e diminuir o hardware**

Pipelines

- **Melhorar o desempenho através do aumento do *throughput***



- **tempo de execução de uma instrução: 8 ns e 9 ns (com ou sem pipe)**
- **throughput: 1 instr / 8ns (sem pipeline) ou 1 instr / 2 ns (com)**
- **# de estágios = 5; ganho de 4:1;**

Pipeline

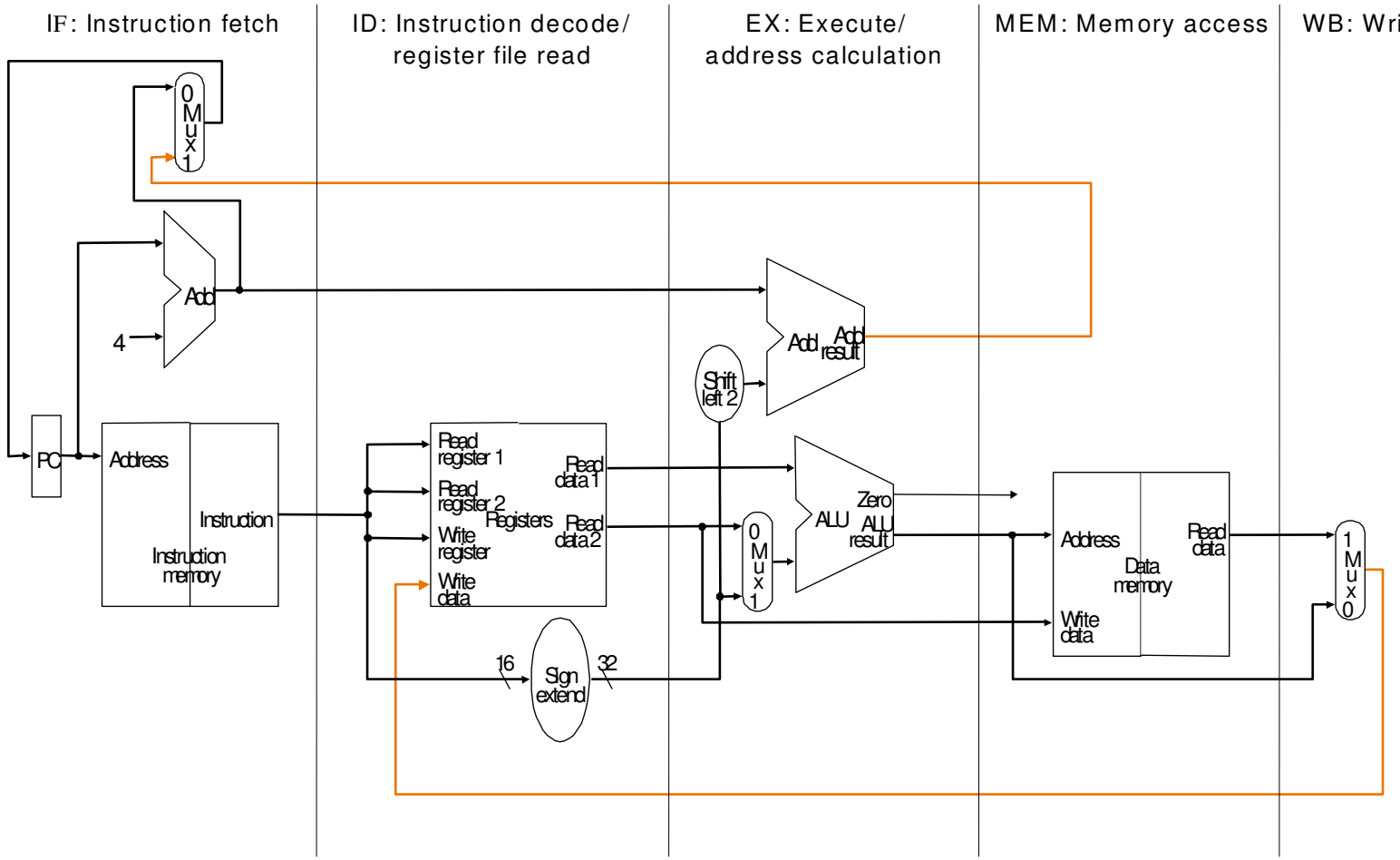
- **O que o torna factível?**
  - **Todas as instruções são do mesmo tamanho**
  - **Poucos formatos de instruções**
  - **Operandos de memória aparecem apenas em *loads* e *stores***

- **O que o torna difícil**
  - ***Hazards* estruturais: recursos compartilhados (Ex: memória)**
  - ***Hazards* de controle: Preocupação com instruções de saltos**
  - ***Hazards* de dados: uma instrução depende dos resultados gerados pela instrução anterior**

- **Nosso foco: um pipeline simples abordando essas questões!**

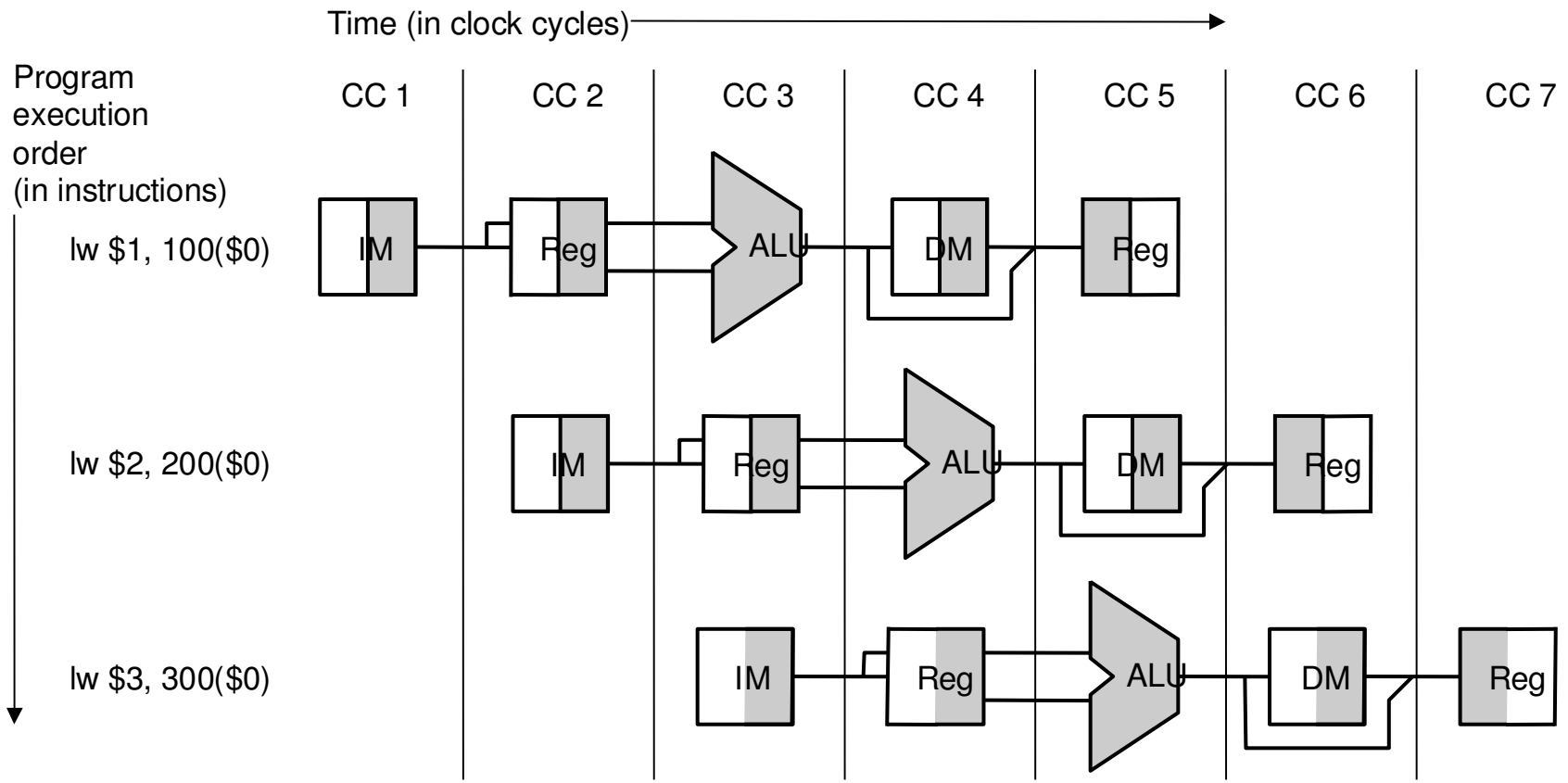
- **Posteriormente (ou durante essa parte do curso..)**
  - **Tratamento de exceções...**
  - **Execução in-order, fora-de-ordem, etc.**

Pipeline: Idéia básica



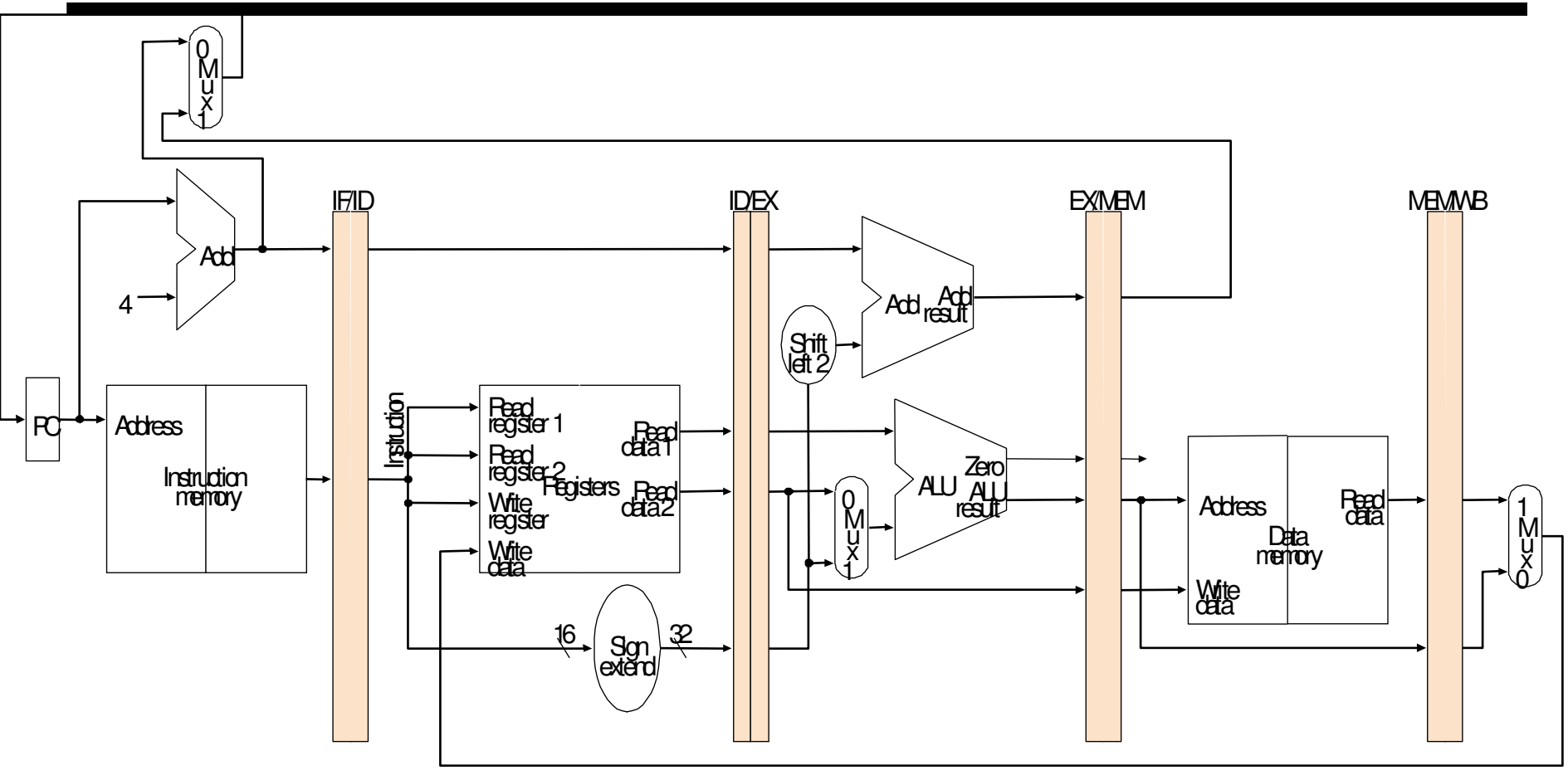
- *O que é necessário para dividir a via de dados em estágios de execução?*

Uma representação para o pipeline



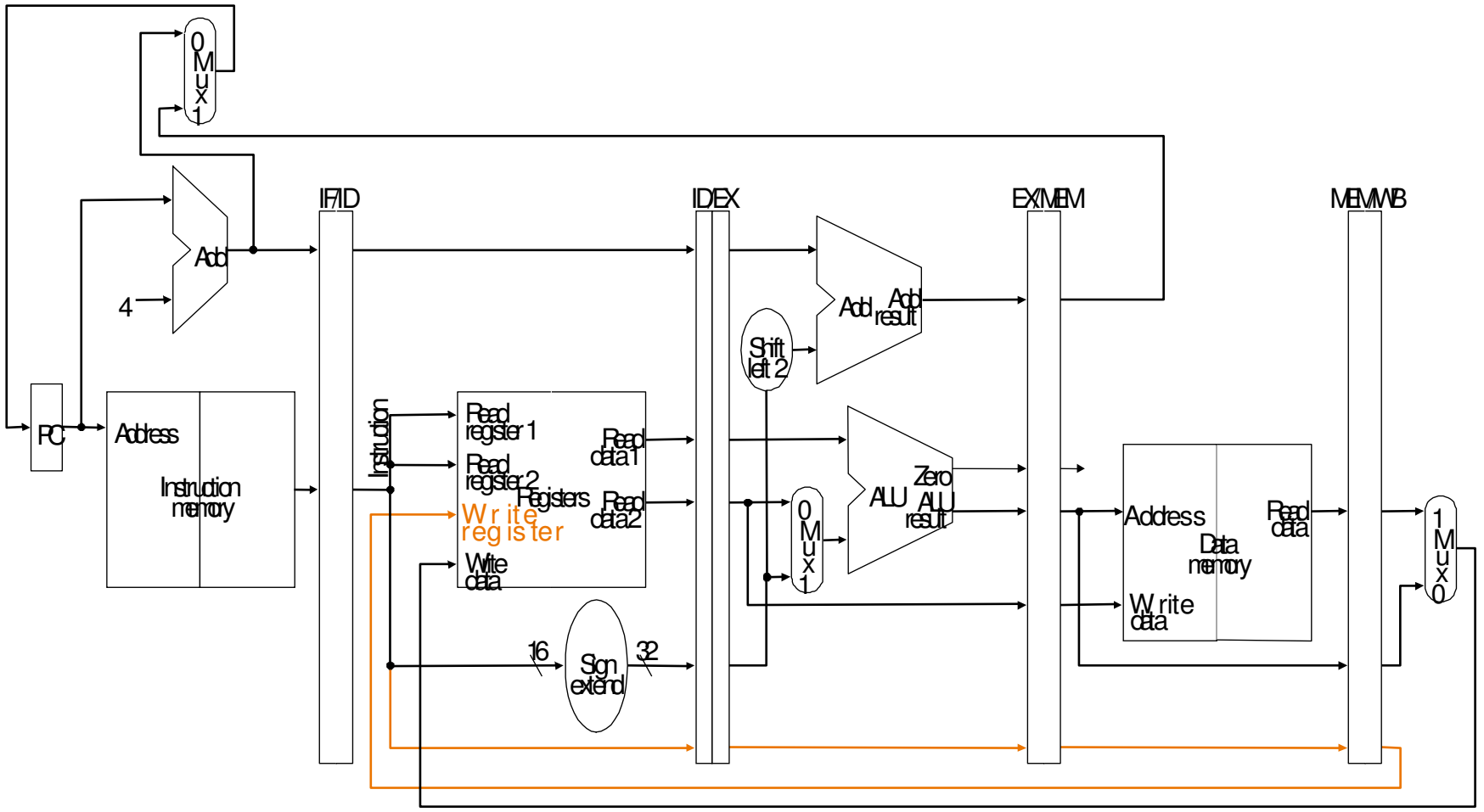
- **Hachurado representa "atividade"**
- **Representação alternativa: imaginando que cada instrução tenha a sua própria via de dados**
- **Outra representação: foto no tempo (*snapshot*)**

Via de dados com pipeline

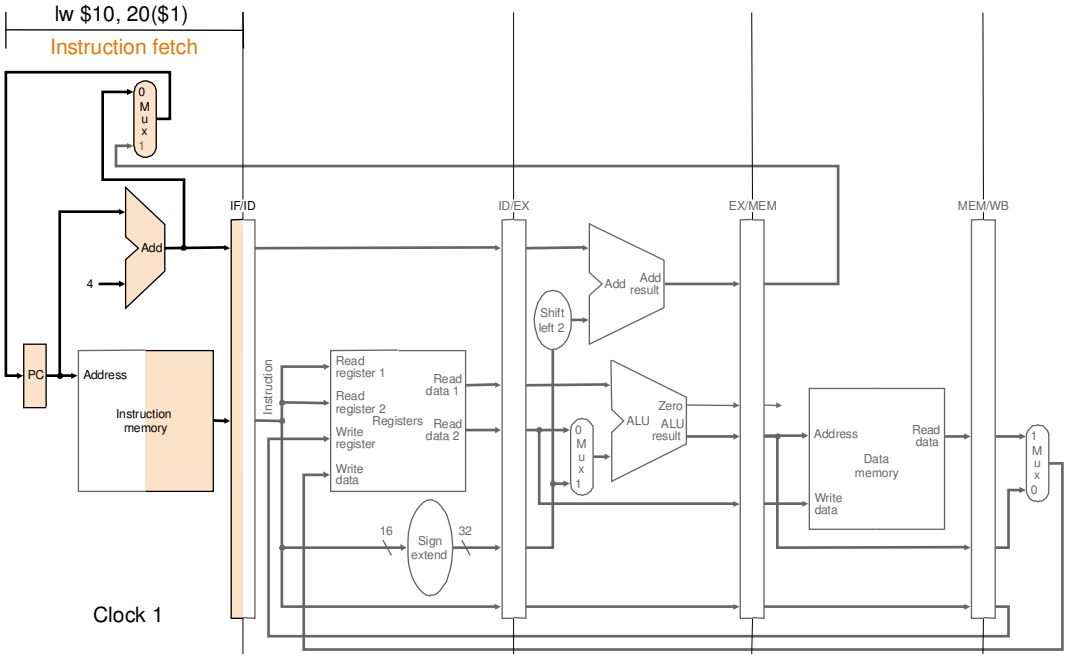


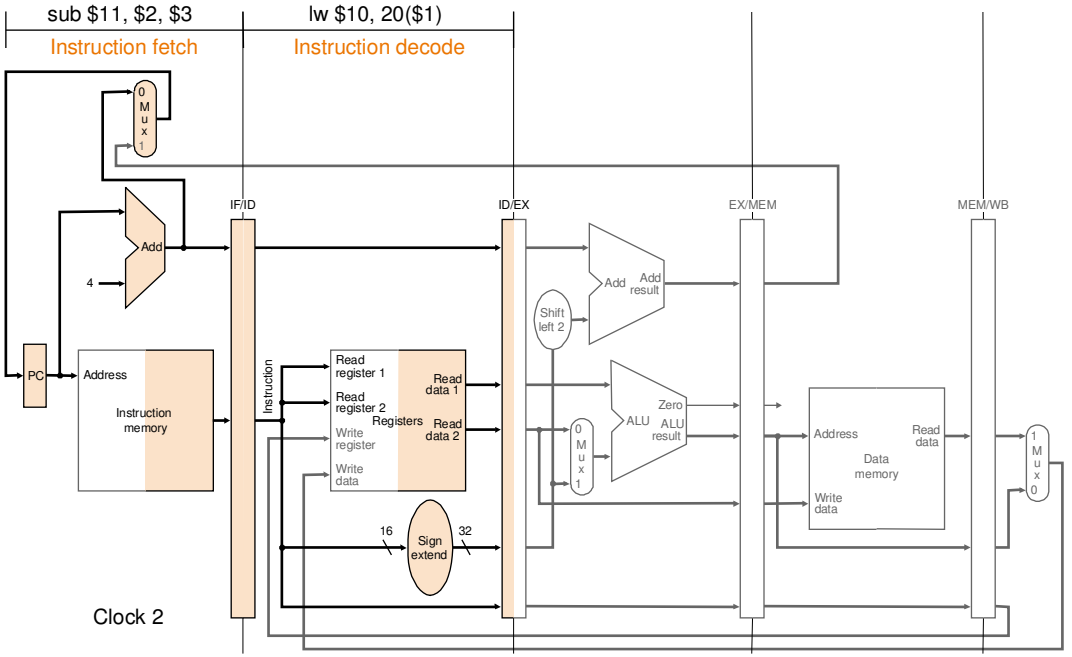
- **O que é produzido em cada estágio é armazenado em um registrador de pipeline para uso pelo próximo estágio**
- **Problemas com o endereço do registrador de escrita?? Caminhando para esquerda?**

Via de dados corrigida

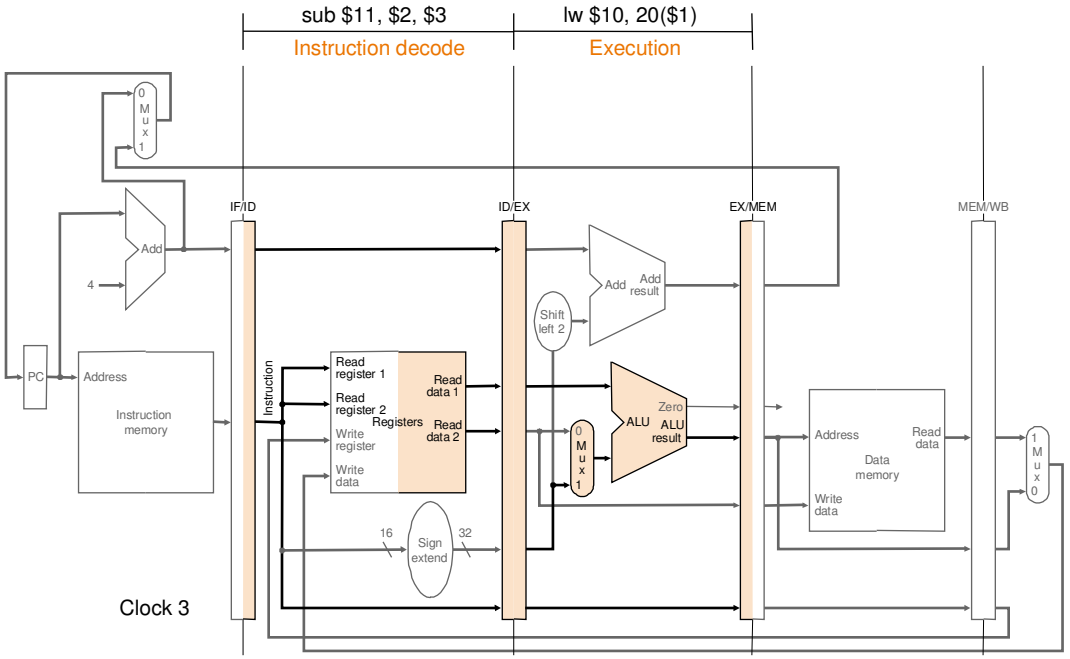


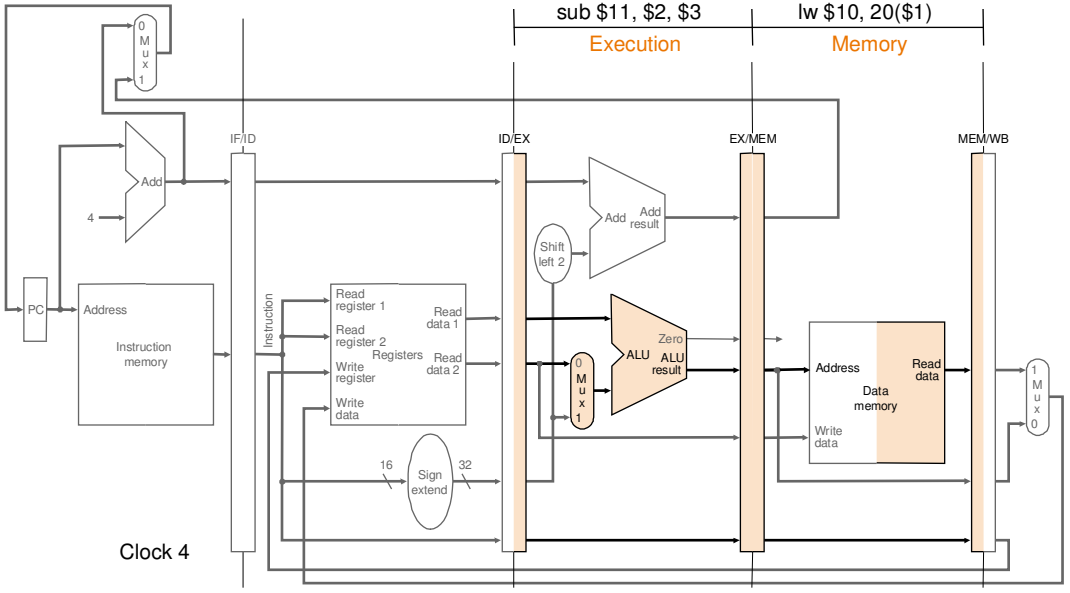
Exemplo com sub e lw (1)



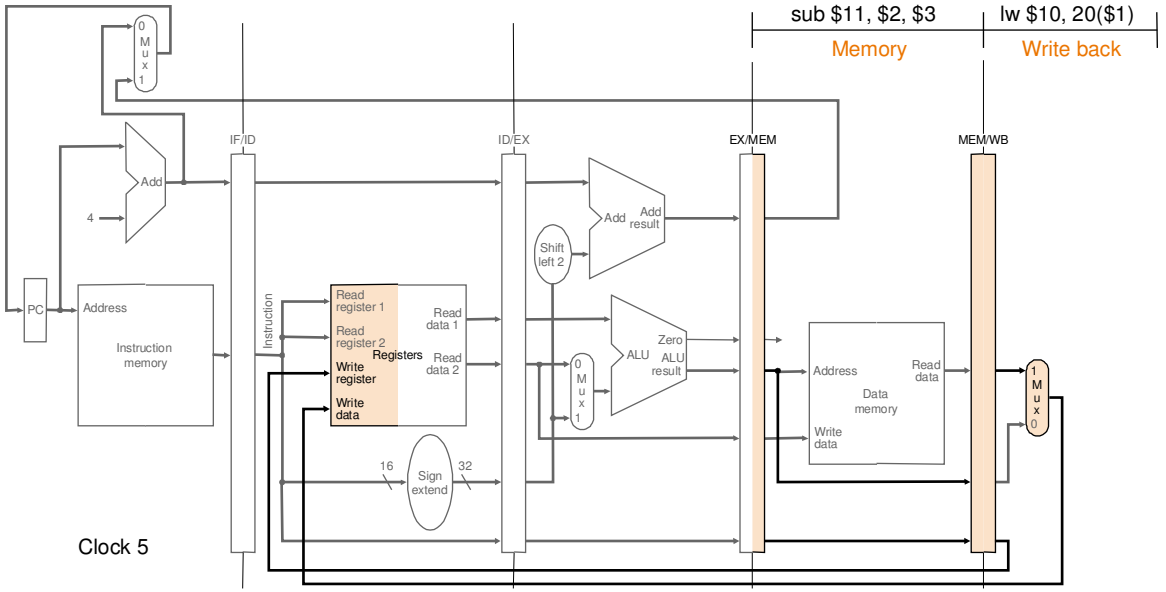


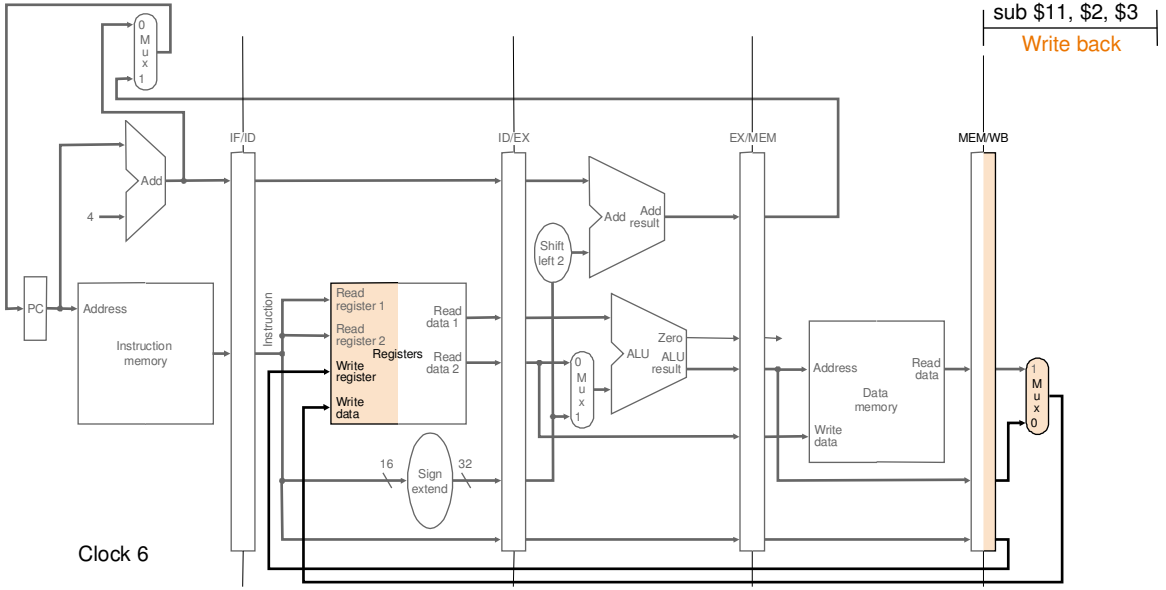
Exemplo com sub e lw (2)



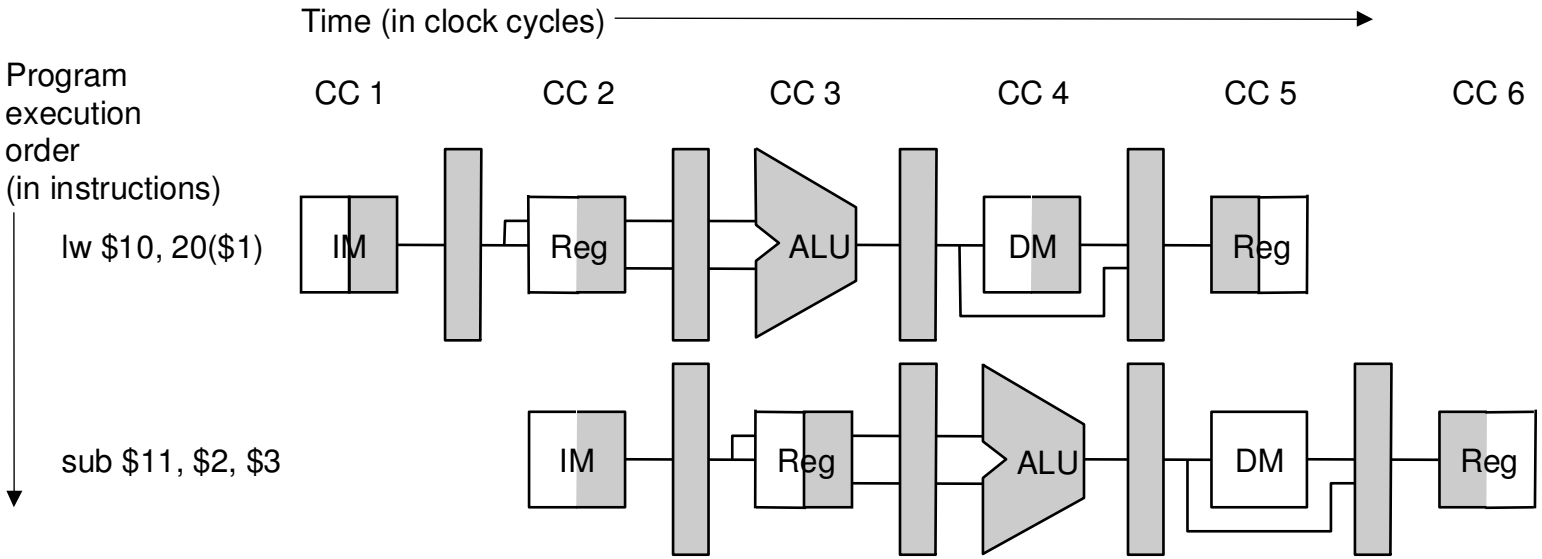


Exemplo com sub e lw (3)





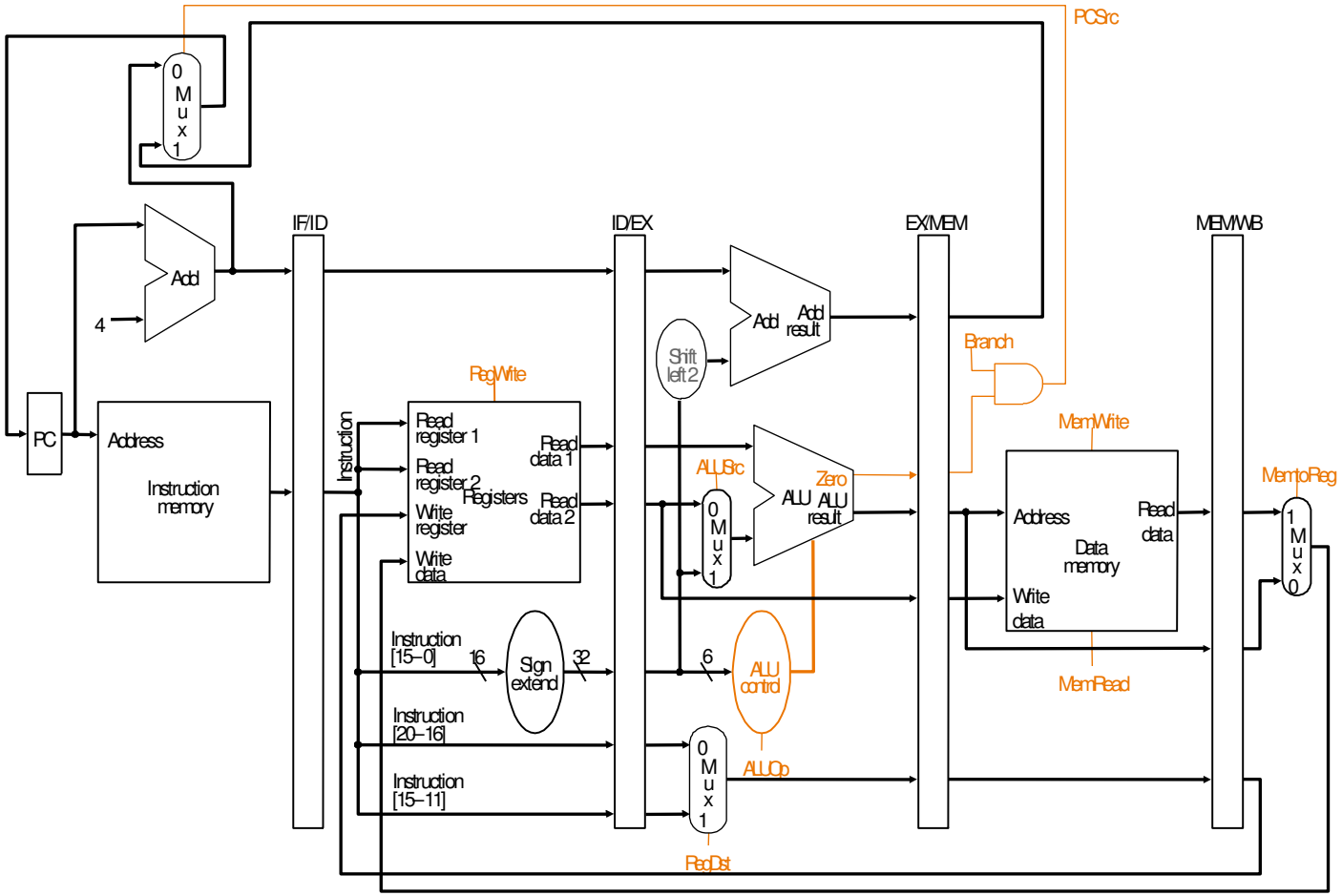
Representando *pipelines* graficamente



- **Essa representação pode ajudar a responder questões como:**
  - **Quantos ciclos são necessários para executar esse código?**
  - **O que a ULA está fazendo no ciclo 4?**

- **Use essa representação para entender melhor como o *pipeline* funciona**

Controle do pipeline



- **campo *function* é usado para controle da ALU (EX)**
- **sinais de controle: os mesmos do capítulo 5, ciclo único**
- **PC e registradores de pipeline escritos em todos ciclos: não é necessário controle**

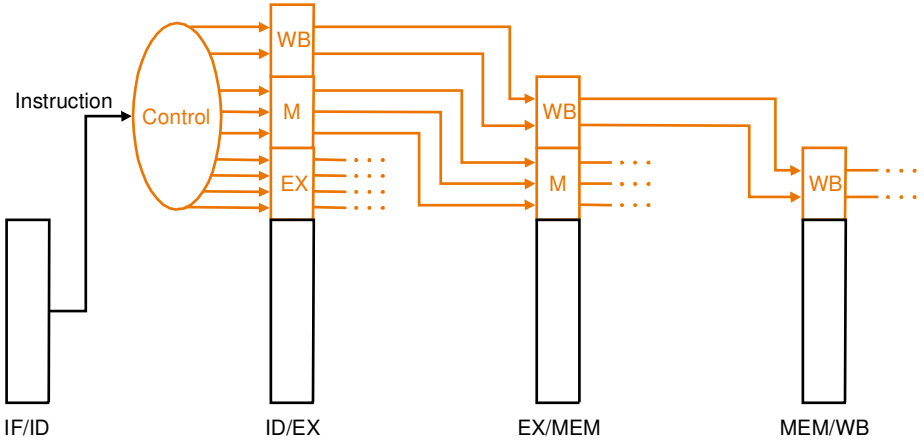
Controle do pipeline

- O que necessita ser controlado em cada estágio?
  - Busca da instrução e incremento do PC
  - Decodificação da instrução / Leitura do registrador
  - Execução
  - Leitura/Escrita da memória
  - Escrita no registrador
- Como o controle seria manipulado em um projeto de automóvel
  - Controle central indicando o que cada parte deve fazer?
  -

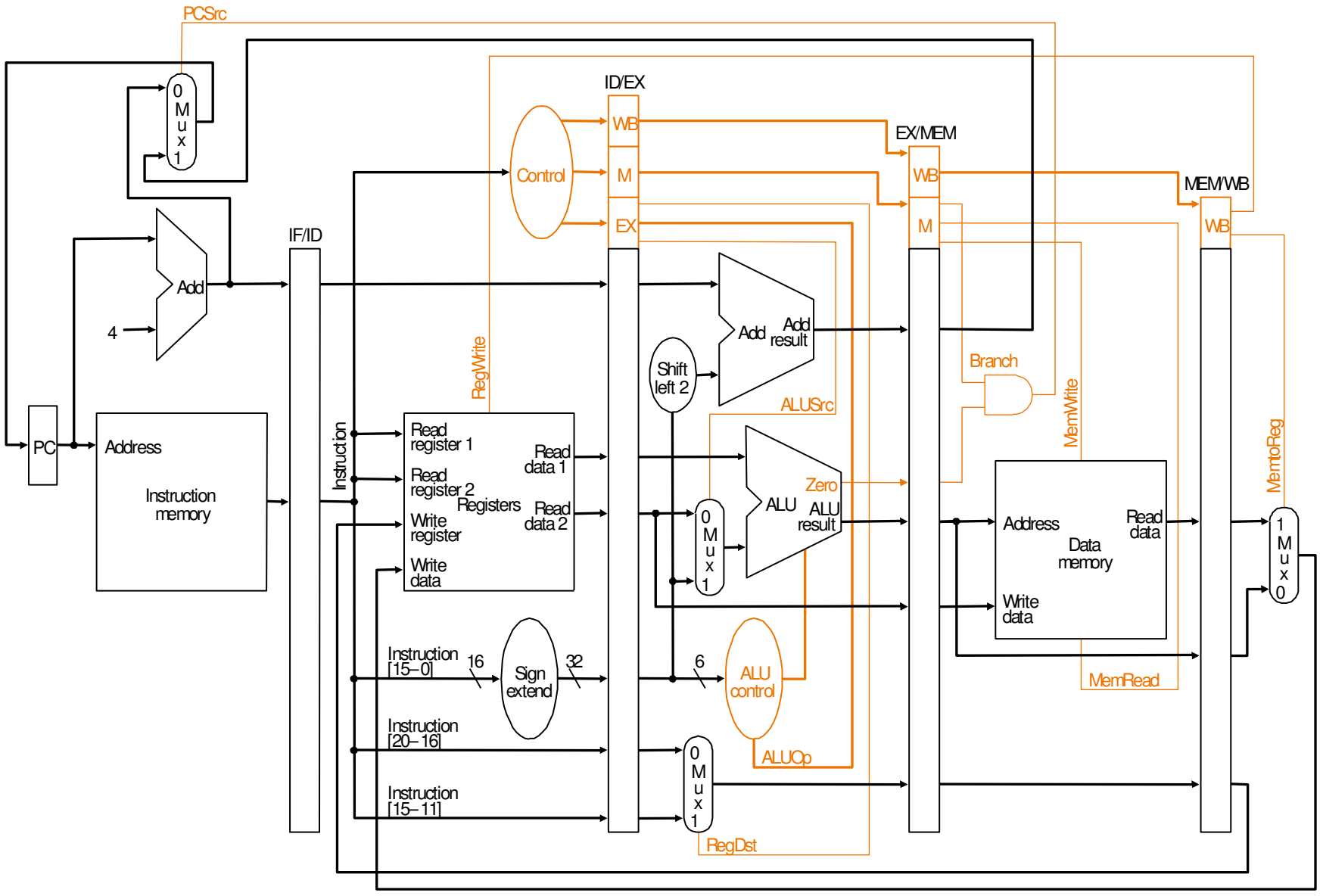
Controle do pipeline

- **Os sinais de controle "passam" pelo pipeline assim como os dados**

Instruction	Execution/Address Calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg write	Mem to Reg
R-format	1	1	0	0	0	0	0	1	0
`lw`	0	0	0	1	0	1	0	1	1
`sw`	X	0	0	1	0	0	1	0	X
`beq`	X	0	1	0	1	0	0	0	X

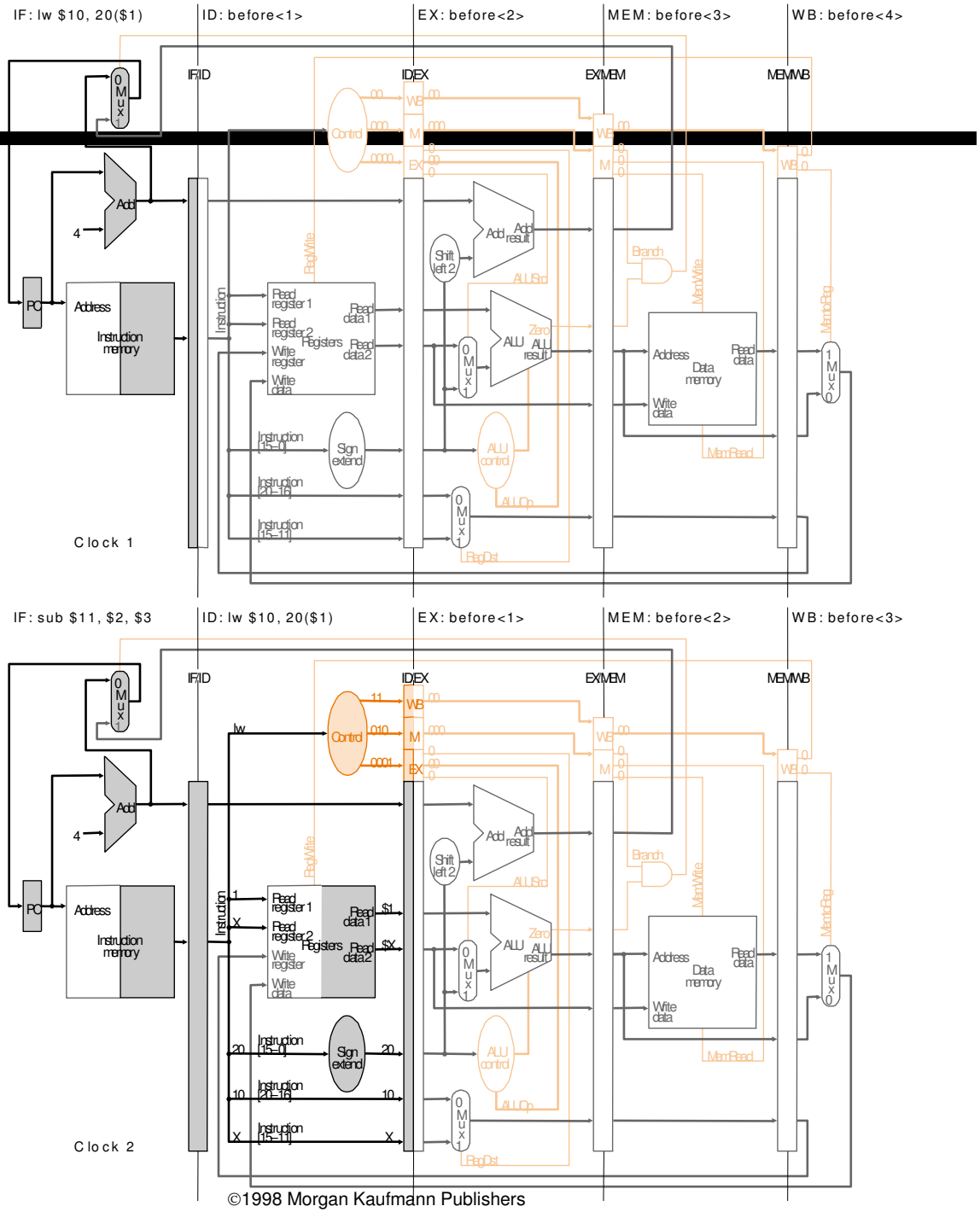


Controle na via de dados



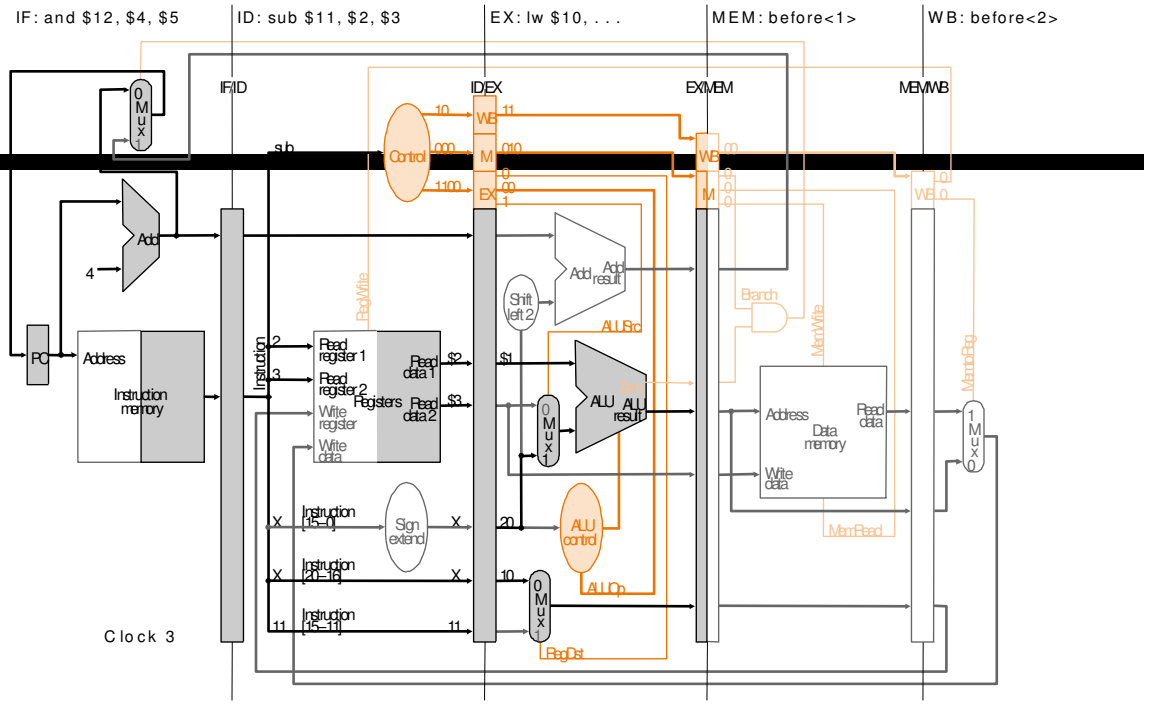
Exemplo (pag. 471)

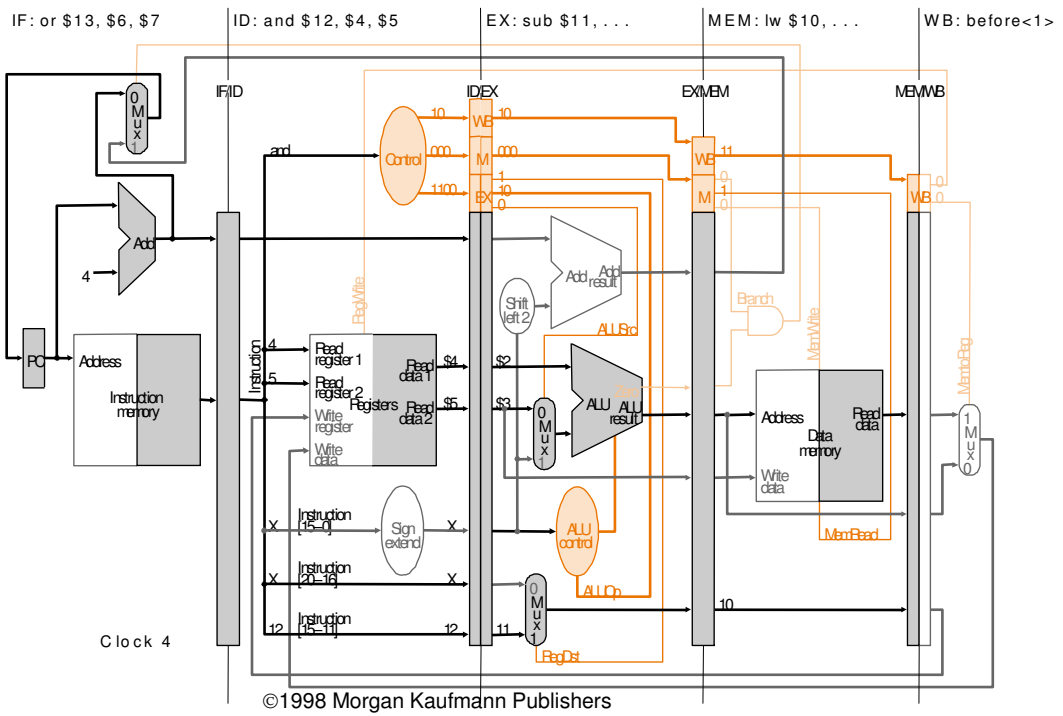
```
lw    $10, 20($1)
sub   $11, $2, $3
and   $12, $4, $5
or    $13, $6, $7
add   $14, $8, $9
```



Exemplo (pag. 471)

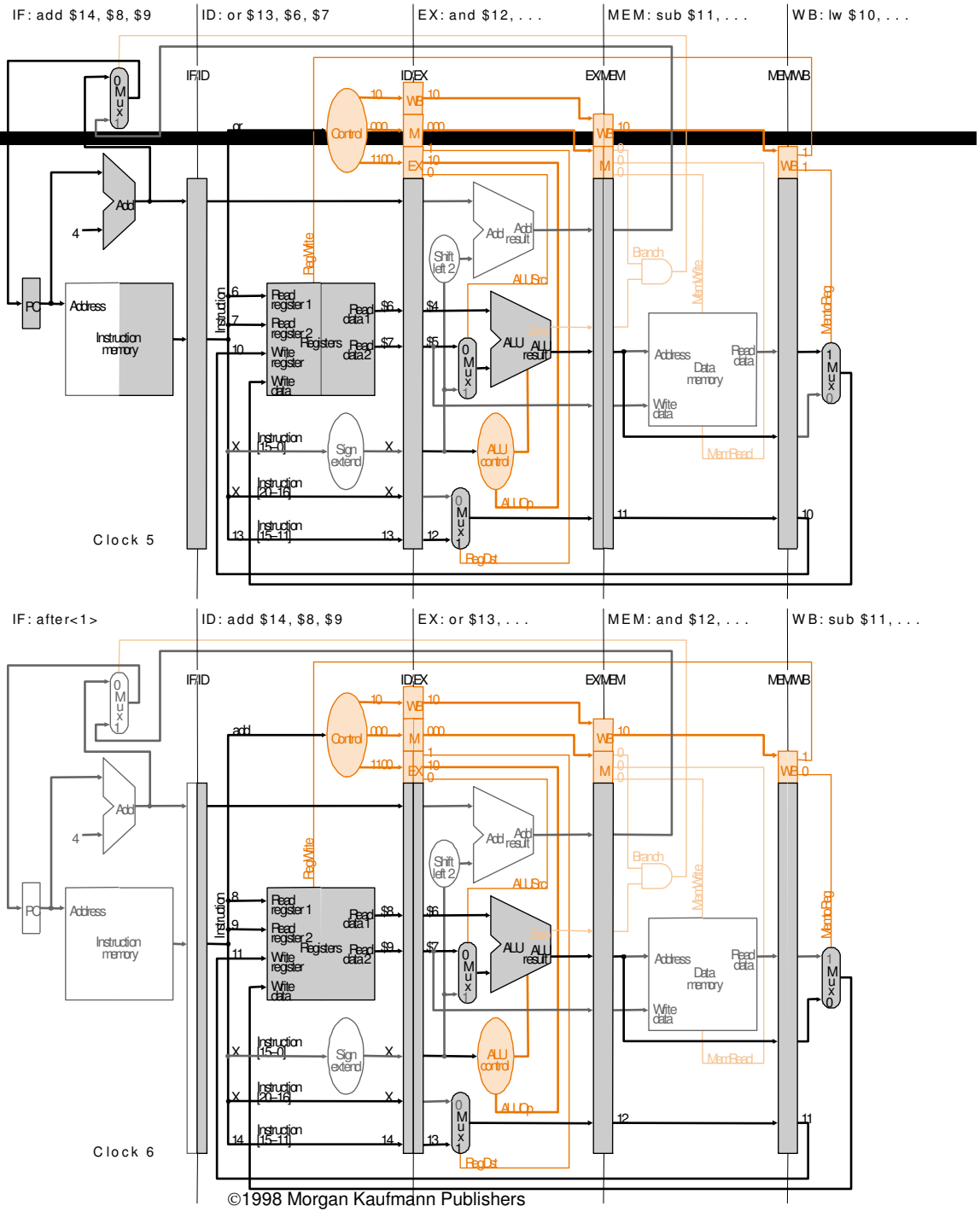
```
lw    $10, 20($1)
sub   $11, $2, $3
and   $12, $4, $5
or    $13, $6, $7
add   $14, $8, $9
```





Exemplo (pag. 471)

```
lw    $10,  20($1)
sub   $11,  $2,  $3
and   $12,  $4,  $5
or    $13,  $6,  $7
add   $14,  $8,  $9
```



Exemplo (pag. 471)

```
lw    $10, 20($1)
sub   $11, $2, $3
and   $12, $4, $5
or    $13, $6, $7
add   $14, $8, $9
```