> “Se você realmente quer aprender, você deve montar a máquina e se acostumar com seus detalhes na prática” (Wilbur Wright).

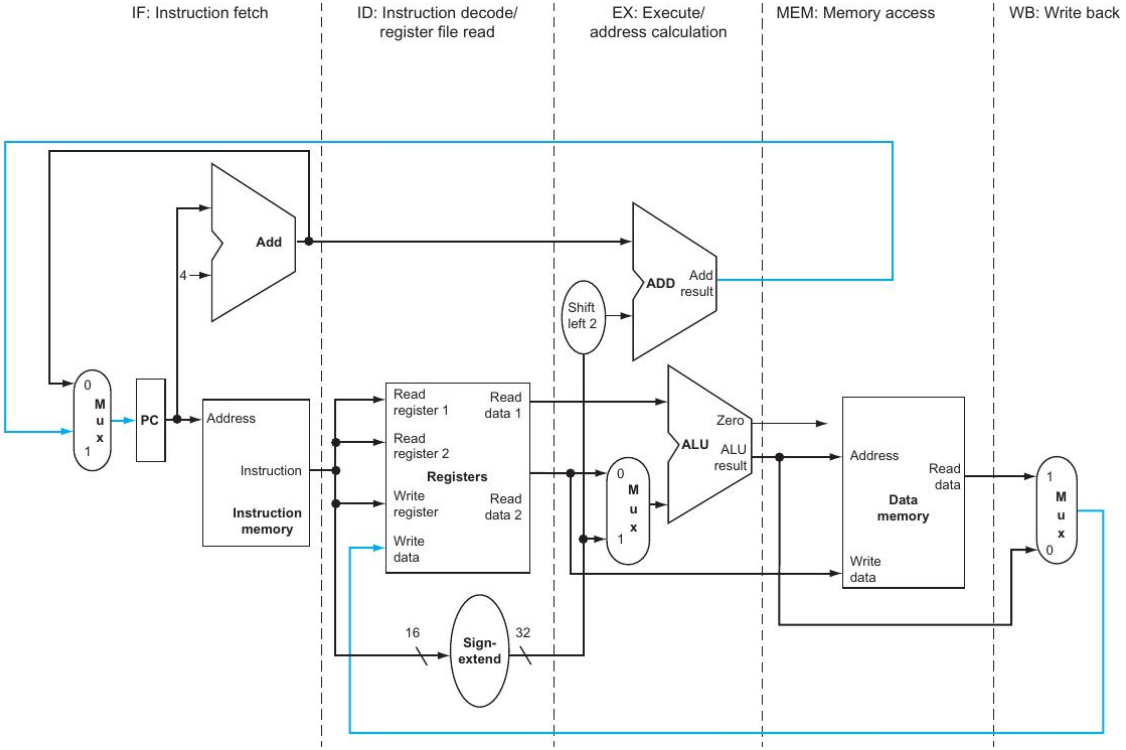
Caminho de dados com pipeline

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

Recapitulando...



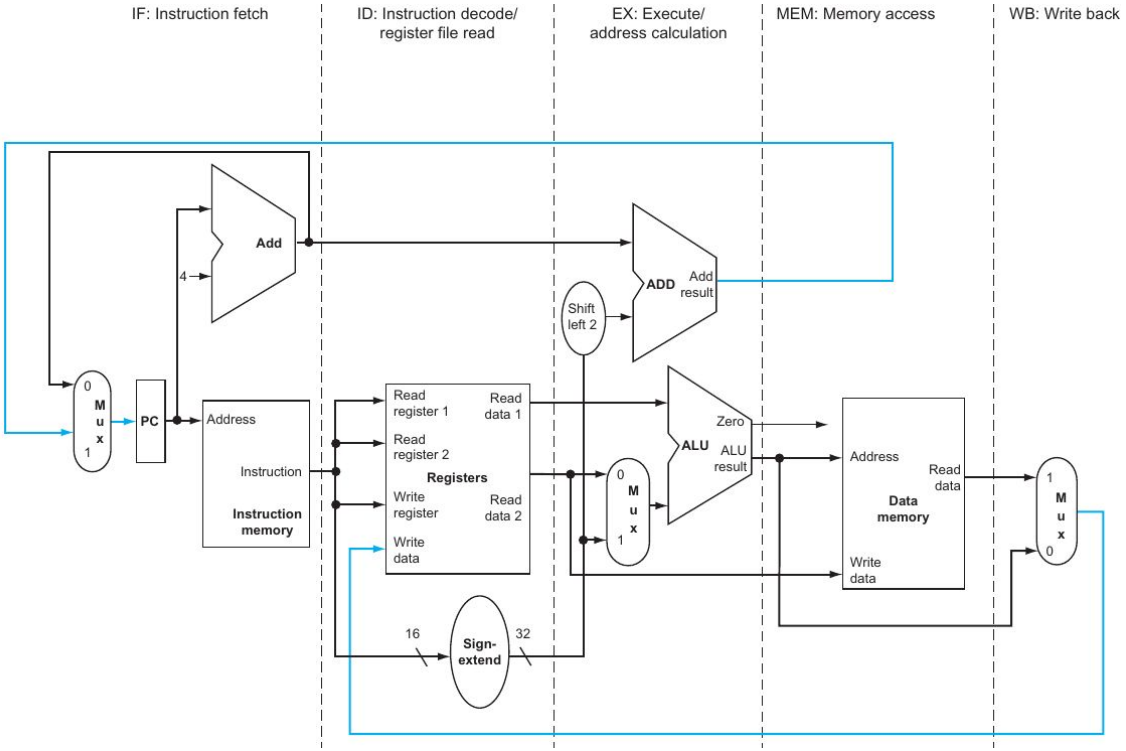
Recapitulando...

Nunca voltamos no tempo.

No geral, os estágios vão da esquerda para a direita, **exceto**:

- Em WB – grava o resultado no registrador.
- A escrita do novo valor de PC.

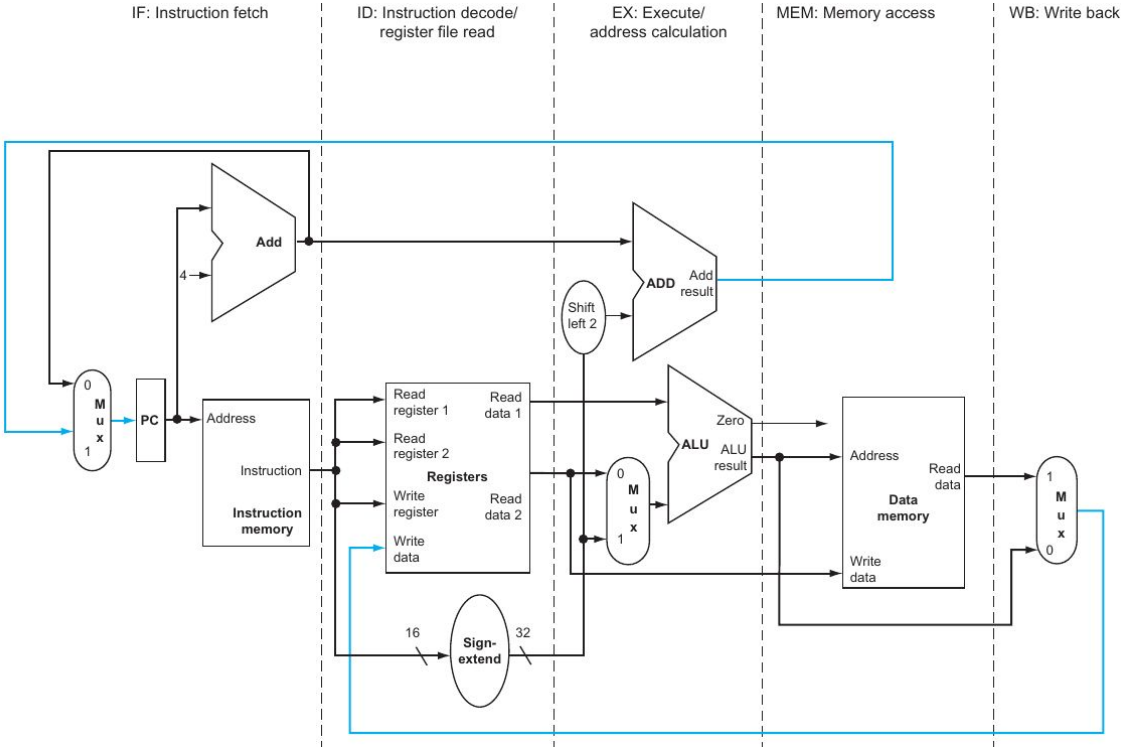
Isso não viola os princípios do pipeline.
Basta visualizar que apesar de alguns componentes desses estágios estarem antes no pipeline, eles são utilizados em estágios posteriores.



Problemas na fronteira

Considere o seguinte programa

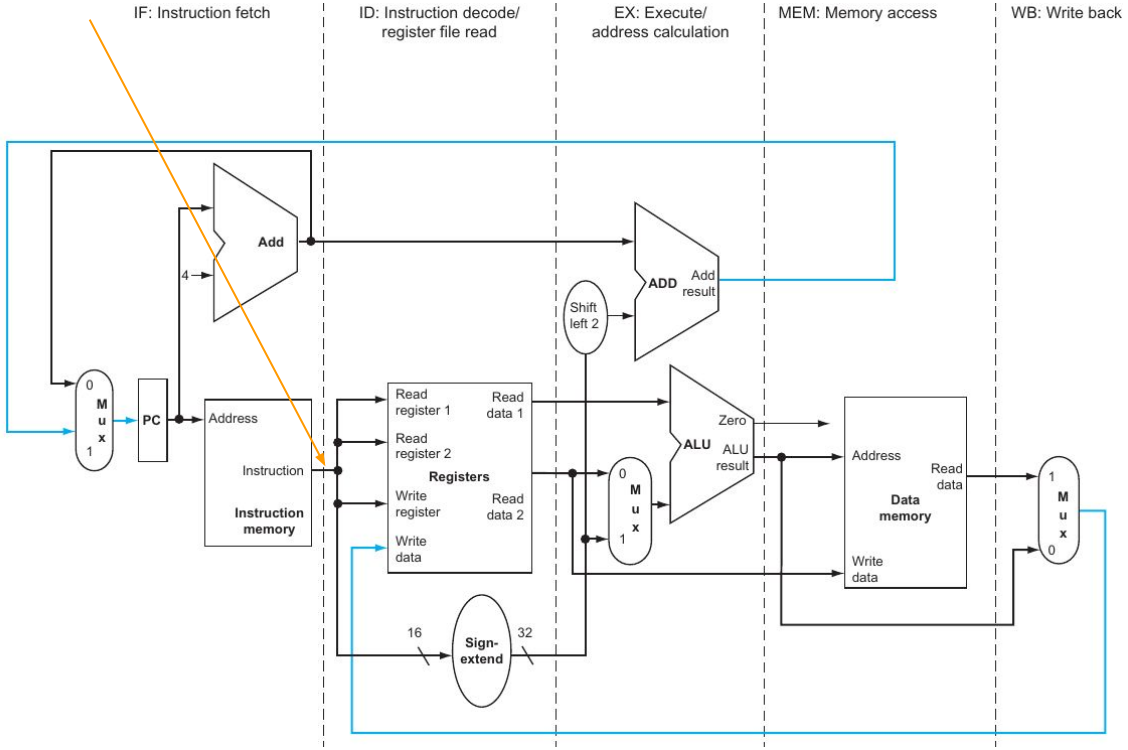
```
lw $1, 100($0)
lw $2, 200($0)
lw $3, 300($0)
```



Problemas na fronteira

pc é enviado no primeiro estágio, de onde vai sair a primeira instrução - lw $s1, 100($s0).

lw $1, 100($0) <- Estágio IF
lw $2, 200($0)
lw $3, 300($0)



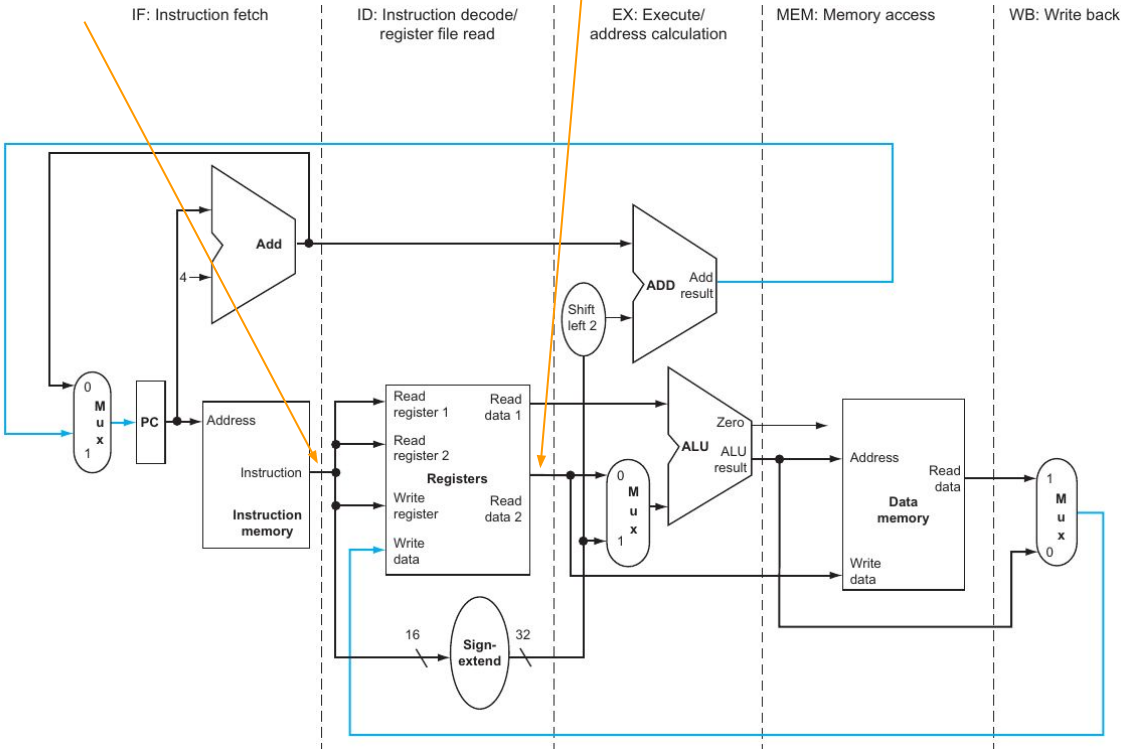
No próximo ciclo...

Problemas na fronteira

lw $s1, 100($s0) é enviado para o estágio ID, para ler os registradores.

pc (atualizado) é enviado no primeiro estágio, de onde vai sair a segunda instrução - lw $s1, 200($s0).

```
lw $1, 100($0) <- Estágio ID
lw $2, 200($0) <- Estágio IF
lw $3, 300($0)
```



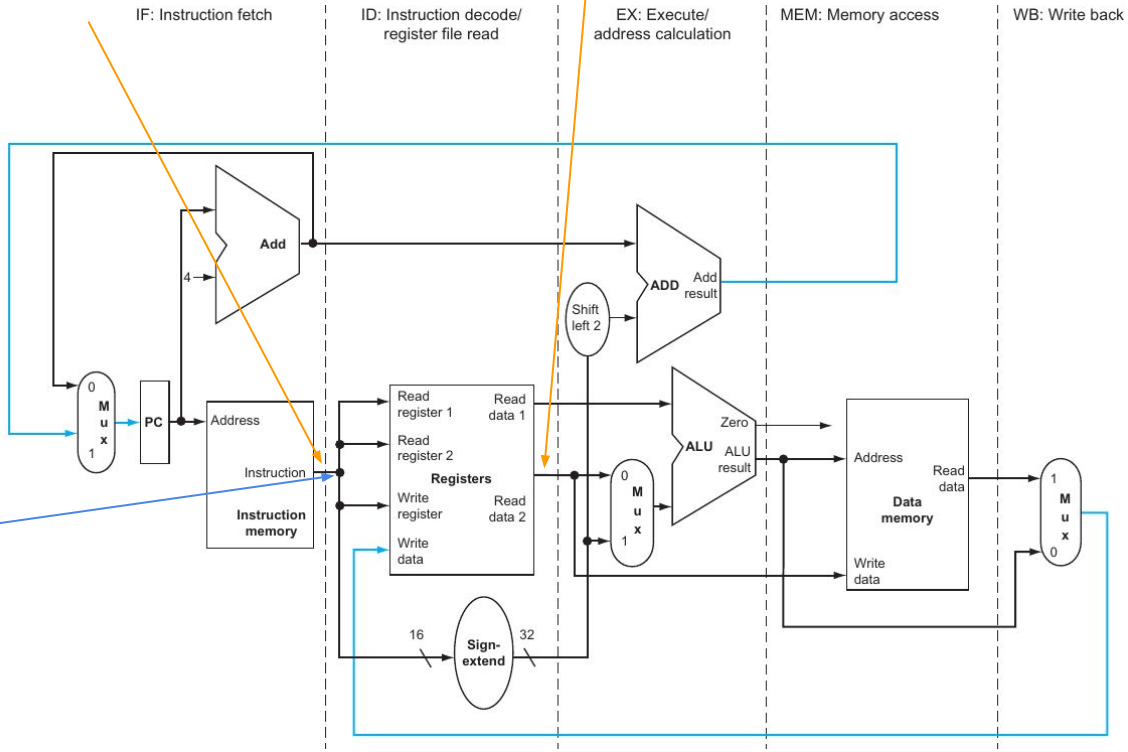
No próximo ciclo...

Problemas na fronteira

lw $s1, 100($s0) é enviado para o estágio ID, para ler os registradores.

pc (atualizado) é enviado no primeiro estágio, de onde vai sair a segunda instrução - lw $s1, 200($s0).

```
lw $1, 100($0) <- Estágio ID
lw $2, 200($0) <- Estágio IF
lw $3, 300($0)
```



Problema: no estágio ID, desejamos a instrução que havia sido carregada no ciclo de clock anterior (lw $1, 100($2)), mas agora só temos o sinal da instrução atual em IF.

Problemas na fronteira

No estágio ID, desejamos a instrução que havia sido carregada no estágio IF no ciclo de clock anterior (lw $1, 100($2)), mas agora só temos o sinal da instrução atual em IF.

O problema se repete nos demais estágios, conforme as instruções “caminham” no pipeline.

Como resolver?

Problemas na fronteira

No próximo ciclo de clock, **o próximo estágio espera continuar o trabalho do estágio anterior**.

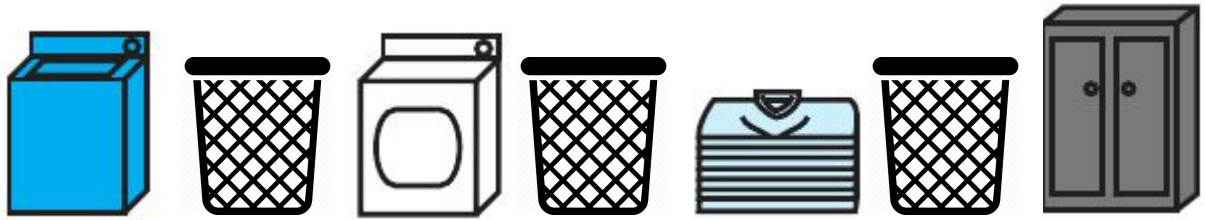
O trabalho foi feito no ciclo de clock anterior.

Necessário salvar o trabalho do estágio anterior.

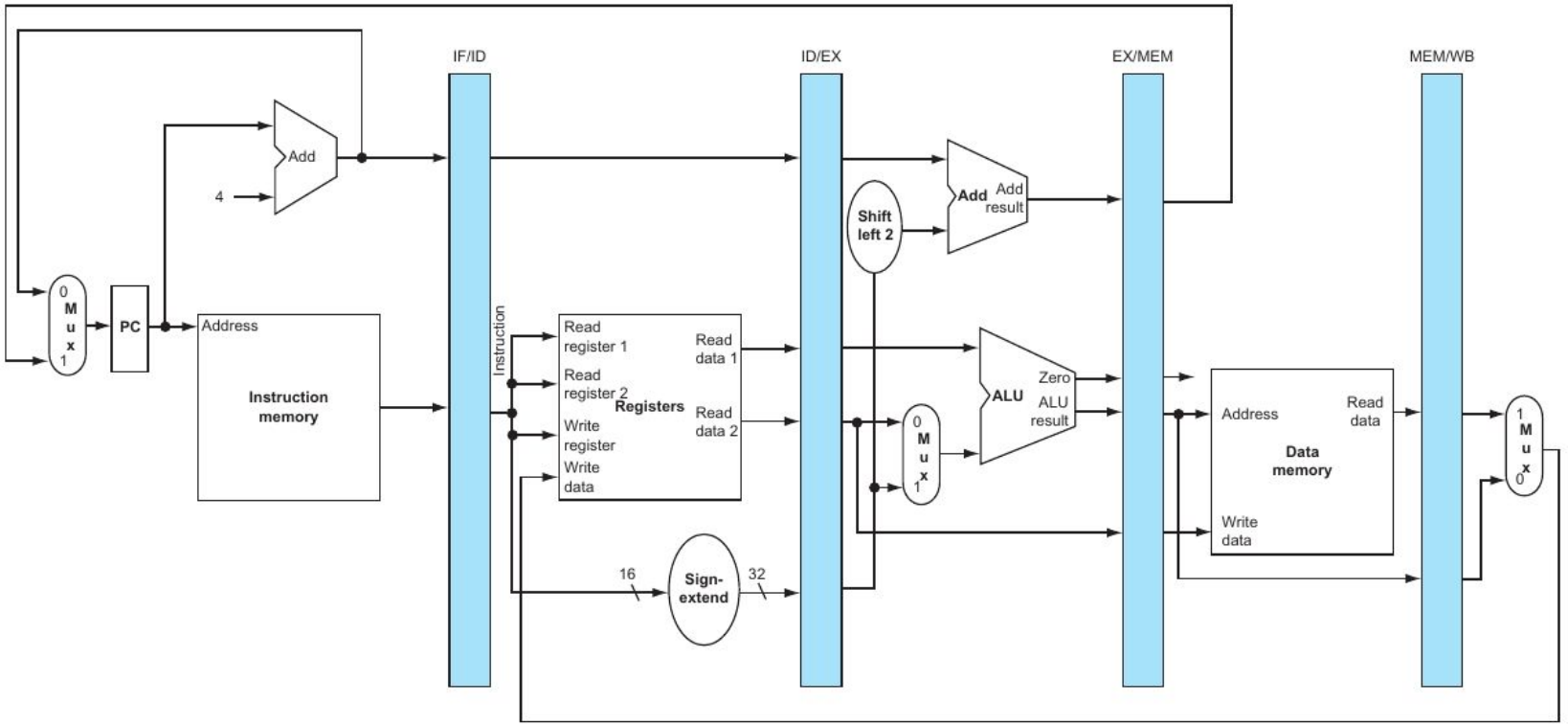
Registradores de pipeline.

Salvar toda a informação que é pertinente para o próximo estágio do pipeline no próximo ciclo de clock.

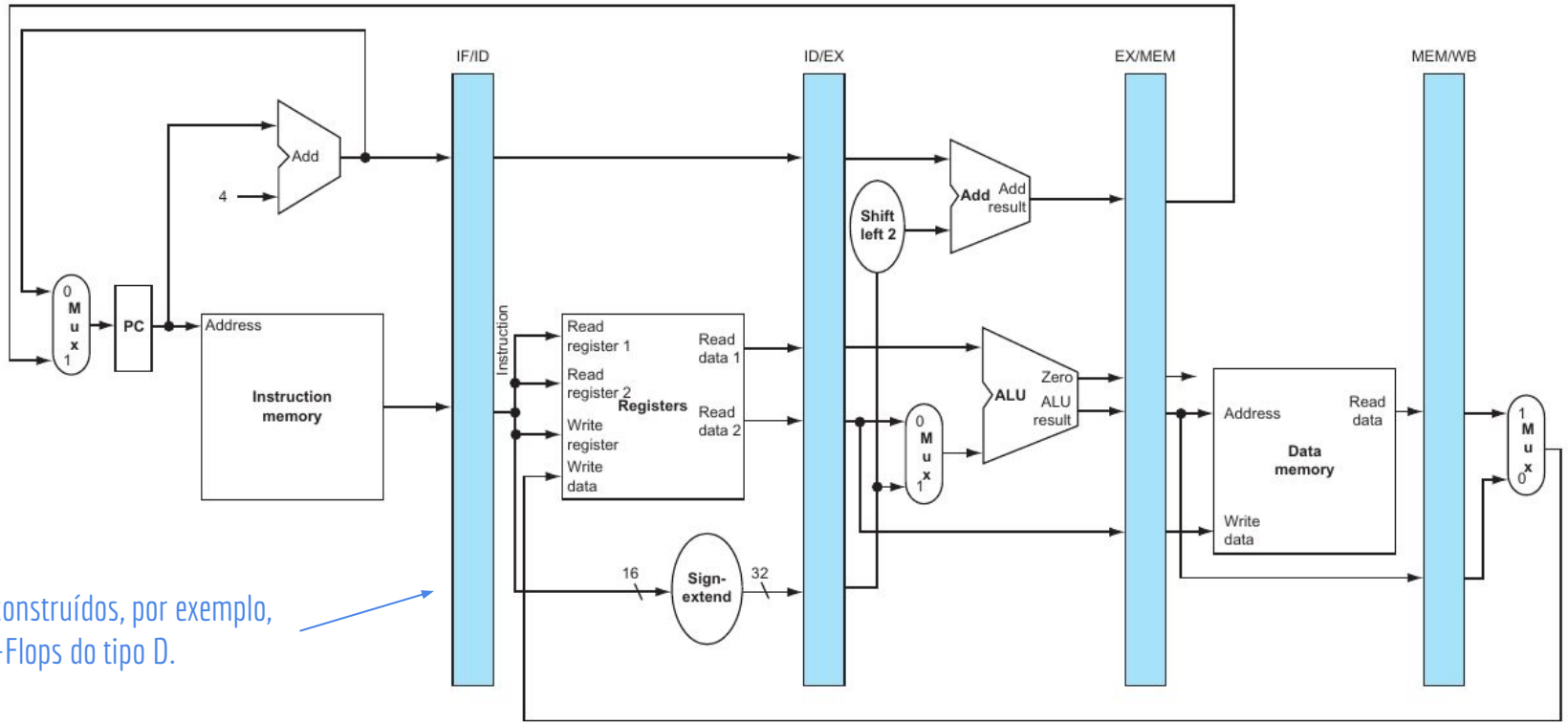
Continuando com o exemplo da lavanderia, teríamos cestos de roupas para armazenar a roupa antes de passar para o próximo estágio.



Registradores de Pipeline (Em azul)



Registradores de Pipeline (Em azul)



Podem ser construídos, por exemplo, usando Flip-Flops do tipo D.

Registradores de Pipeline

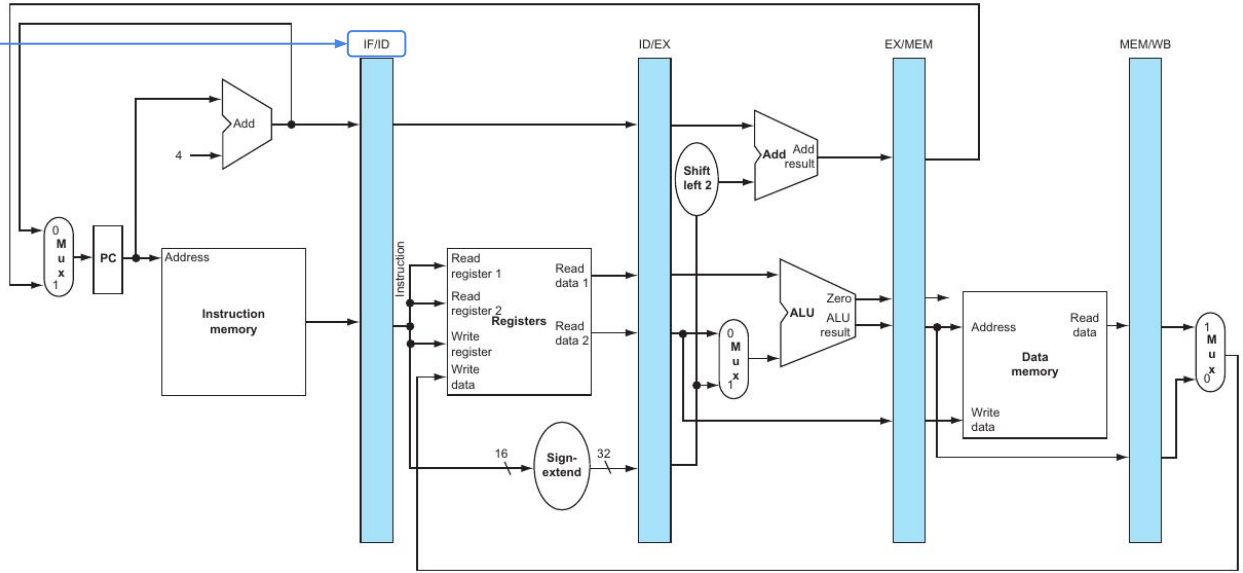
Os registradores que separam o estágio *i* do estágio *j*, são chamados registradores *i/j*.

Exemplo: registradores IF/ID.

Os registradores **IF/ID** precisam armazenar pelo menos 64 bits (32 do PC+4, e 32 da instrução)

Considerando o circuito atual, quantos bits possuem os demais registradores de pipeline?

ID/EX: ?
EX/MEM: ?
MEM/WB:?



Registradores de Pipeline

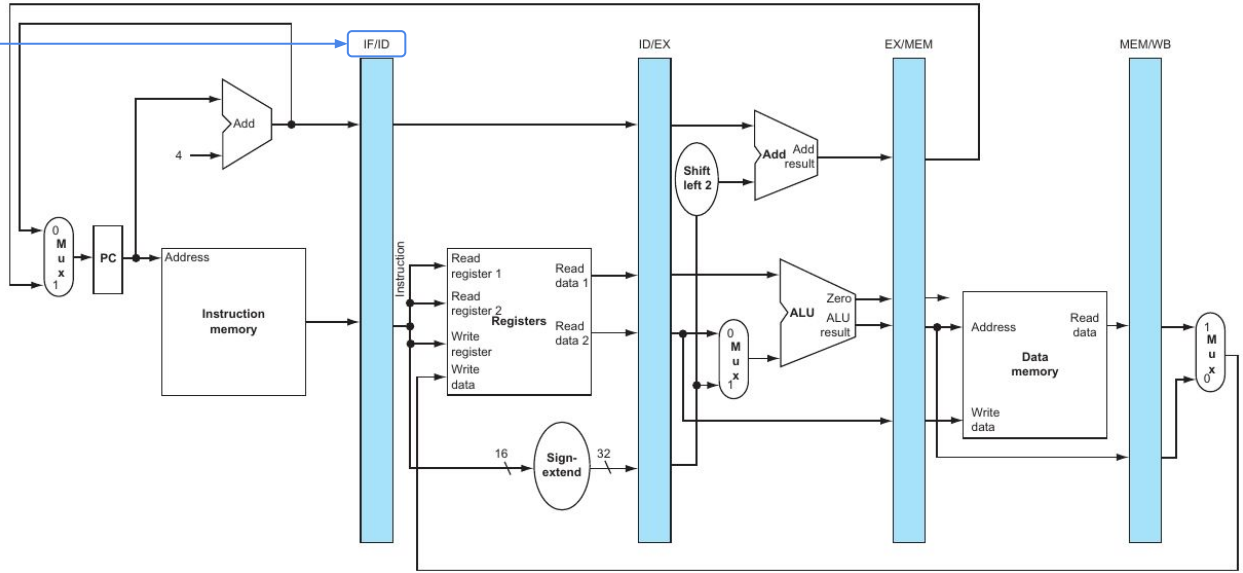
Os registradores que separam o estágio i do estágio j, são chamados registradores i/j

Exemplo: registradores IF/ID.

Os registradores **IF/ID** precisam armazenar pelo menos 64 bits (32 do PC+4, e 32 da instrução)

Considerando o circuito atual, quantos bits possuem os demais registradores de pipeline?

ID/EX: 128 bits
EX/MEM: 97 bits
MEM/WB: 64 bits



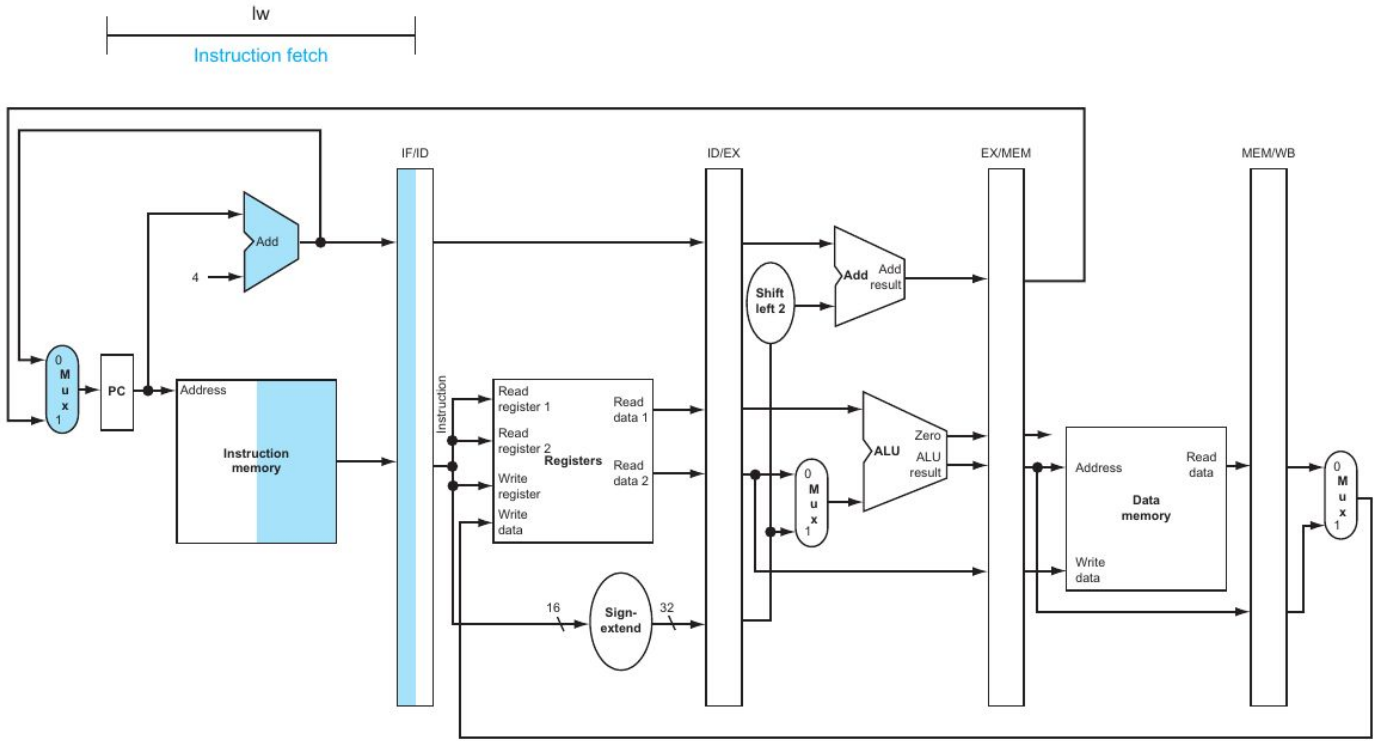
Exemplo

Fluxo de uma instrução lw no pipeline.

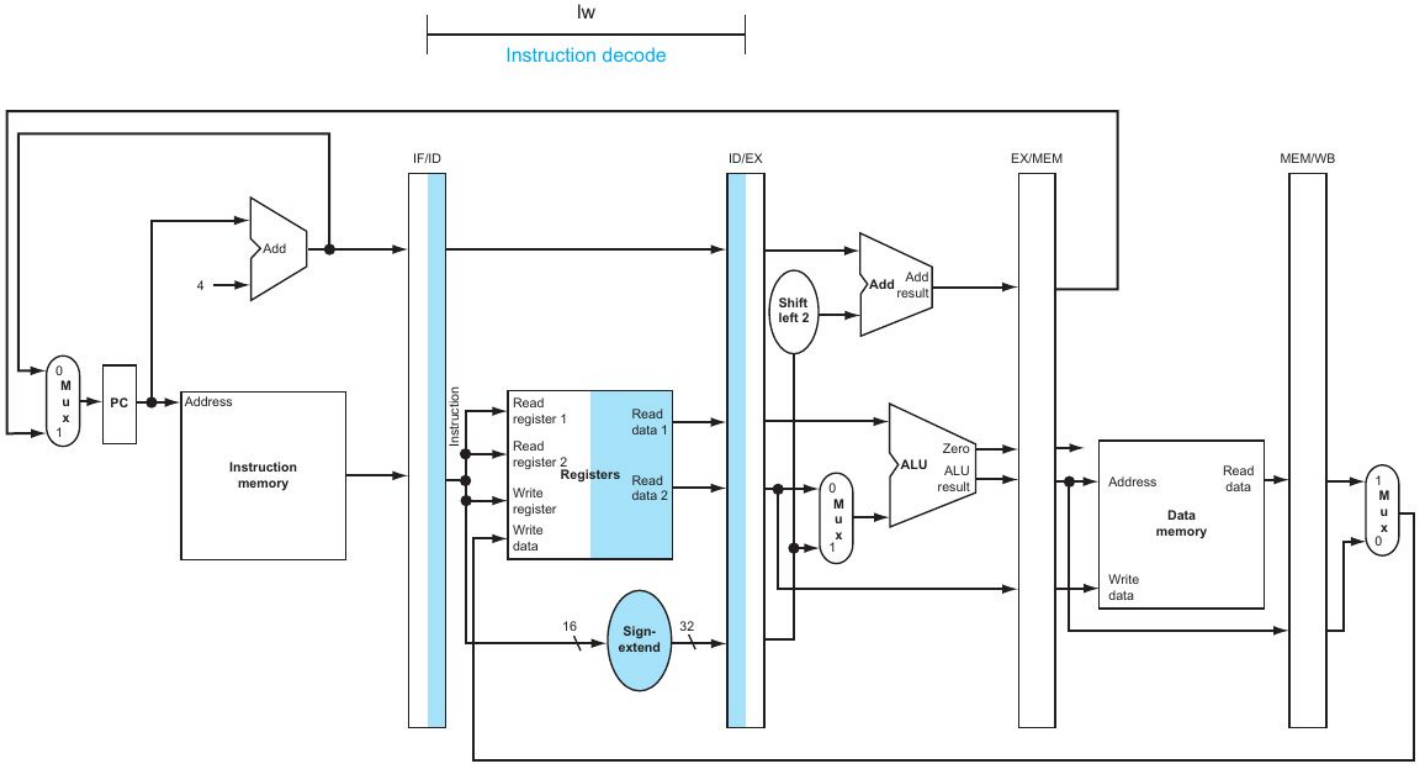
Considere que:

Quando a área sombreada dos registradores é a esquerda, os registradores estão sendo escritos;
Quando a área sombreada dos registradores é a direita, os registradores estão sendo lidos.

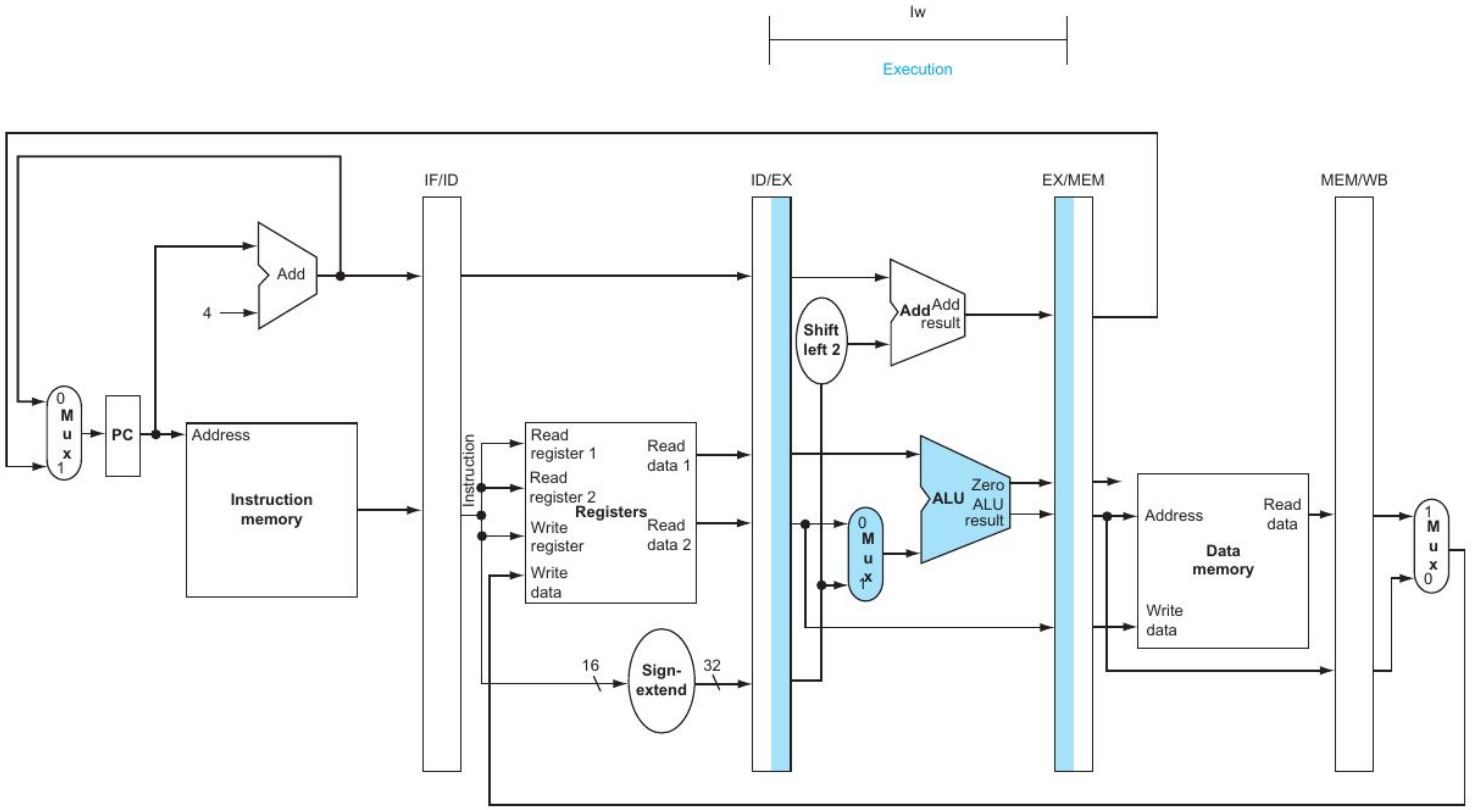
lw no Pipeline



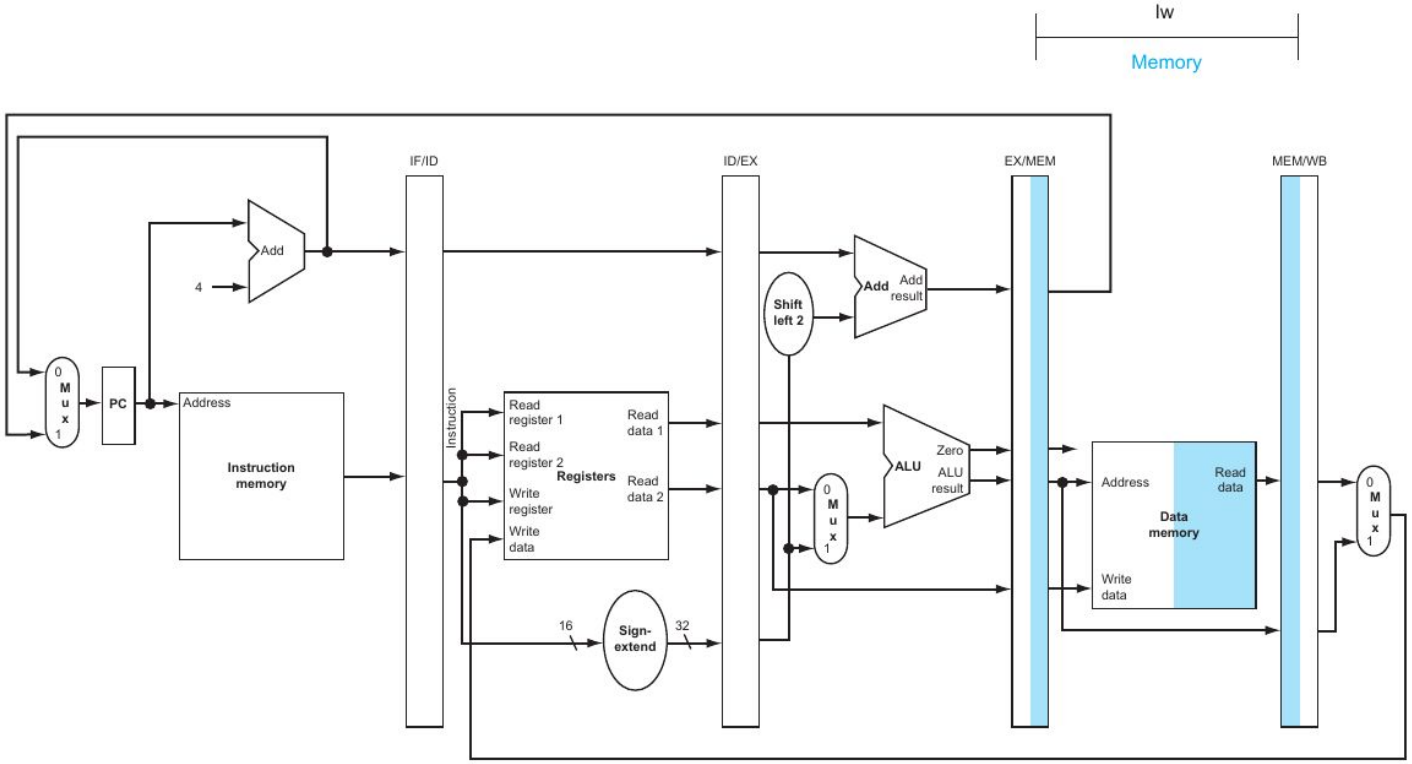
lw no Pipeline



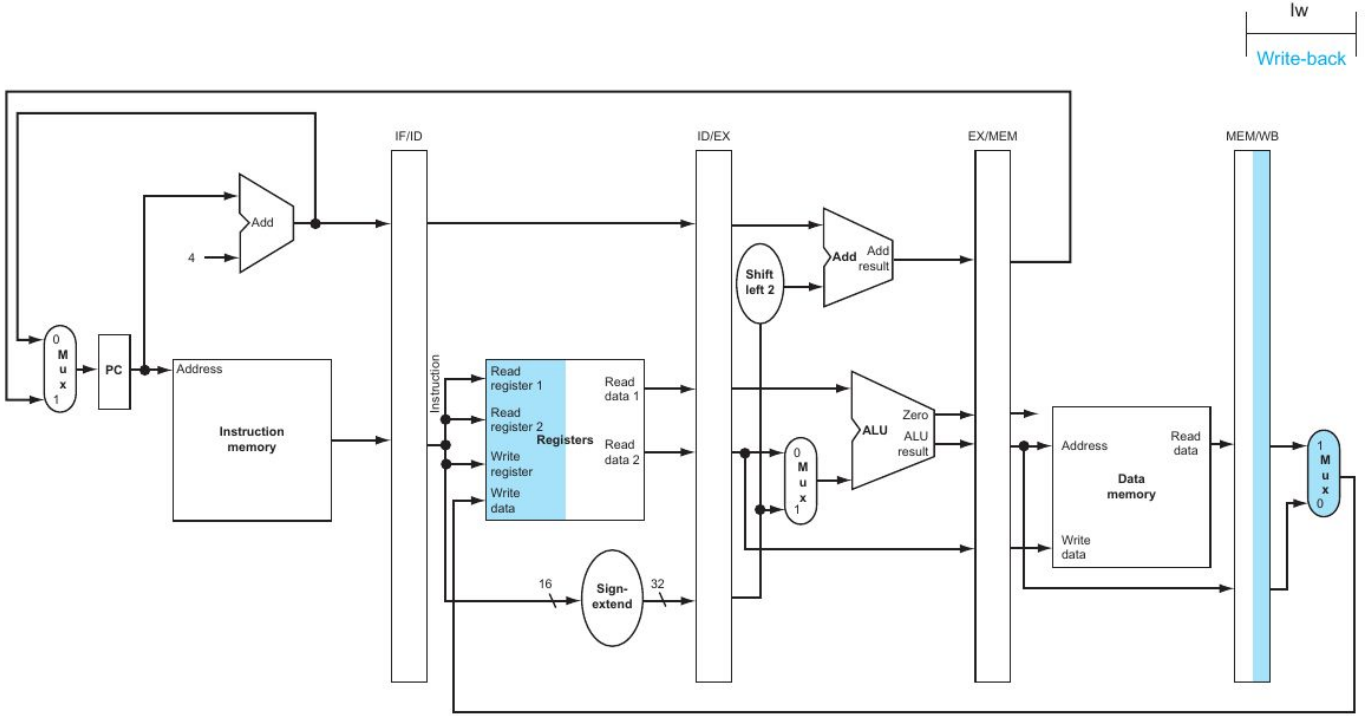
lw no Pipeline



lw no Pipeline

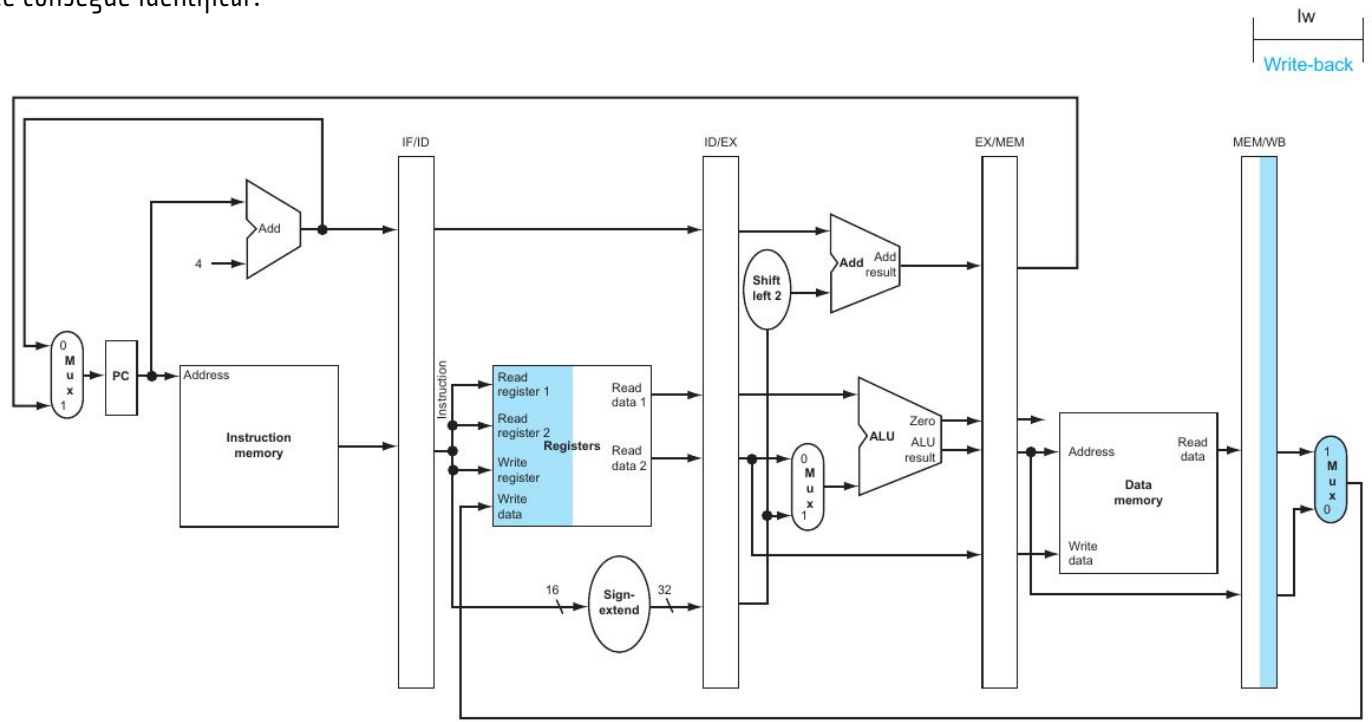


lw no Pipeline



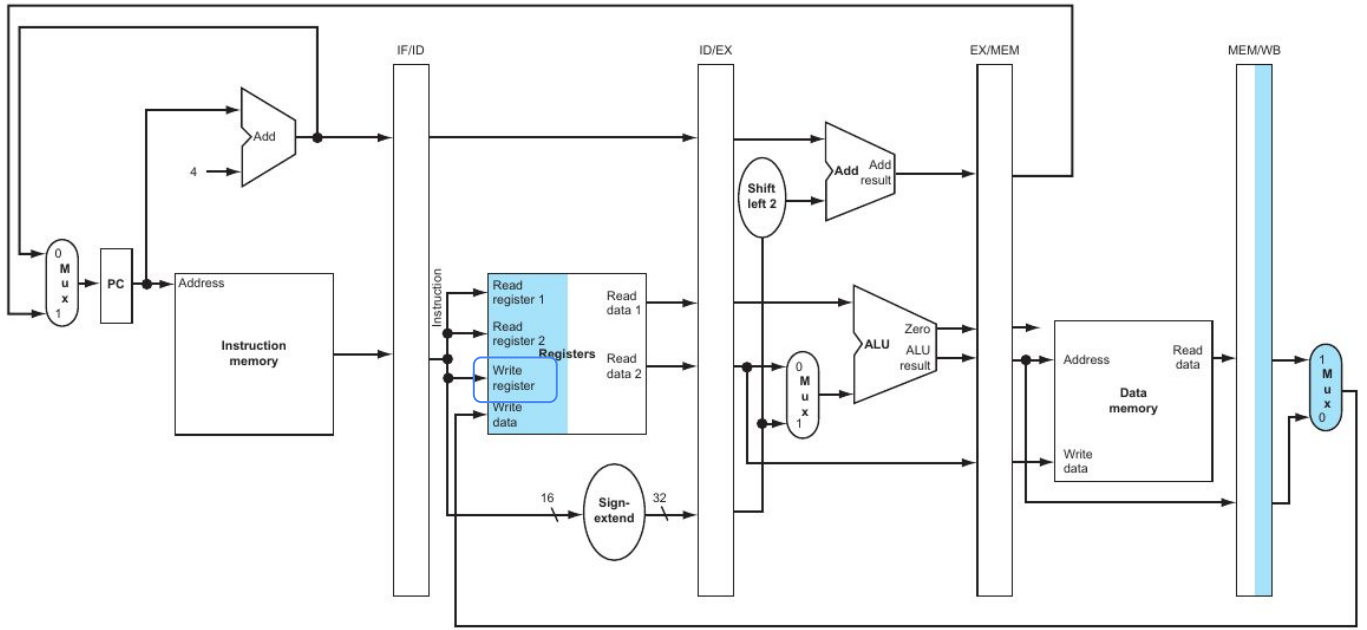
BUG

Já temos um BUG. Você consegue identificar?



BUG

Necessário salvar o endereço do registrador de escrita até o estágio WB. Esse valor está se perdendo no pipeline, e estamos escrevendo no registrador endereçado pela instrução que se encontra no estágio ID, e não pela instrução do estágio WB.

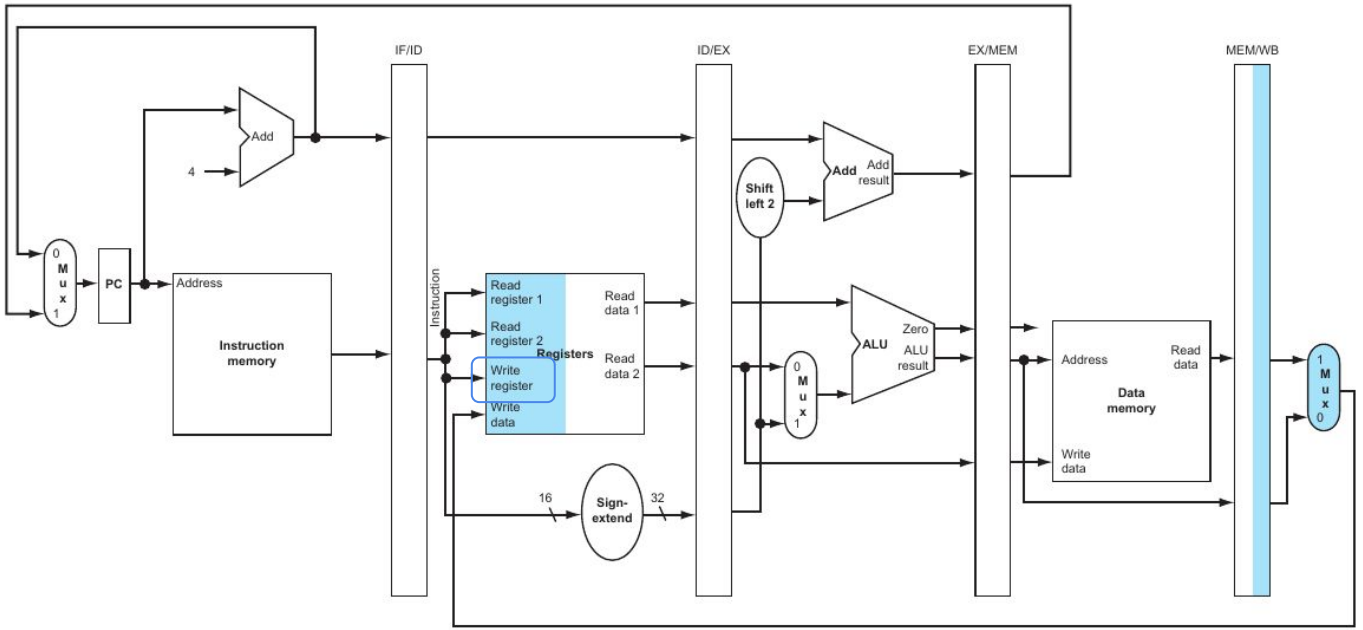


BUG

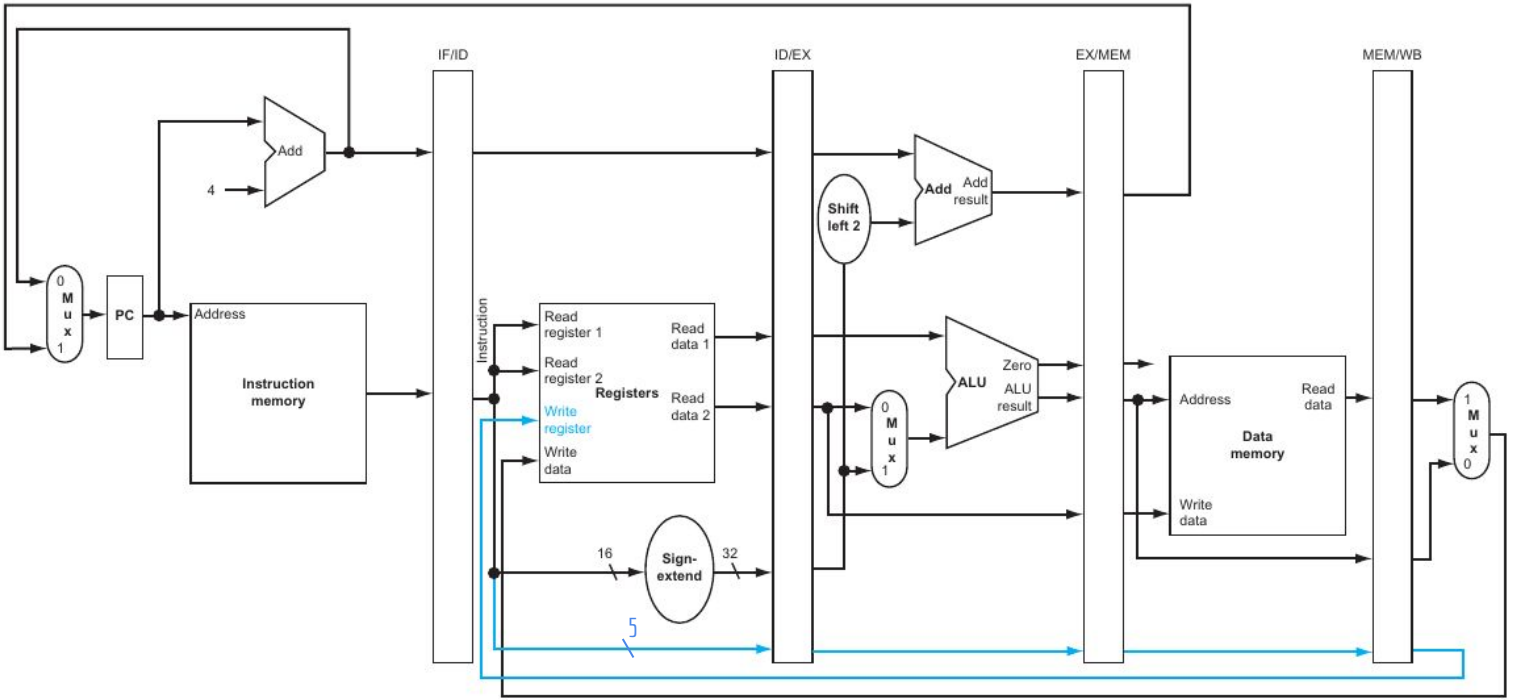
Necessário salvar o endereço do registrador de escrita até o estágio WB. Esse valor está se perdendo no pipeline, e estamos escrevendo no registrador endereçado pela instrução que se encontra no estágio ID, e não pela instrução do estágio WB.

Faça você mesmo.

Corrija o bug, e informe os tamanhos em bits dos registradores de pipeline após a correção.



Correção do BUG

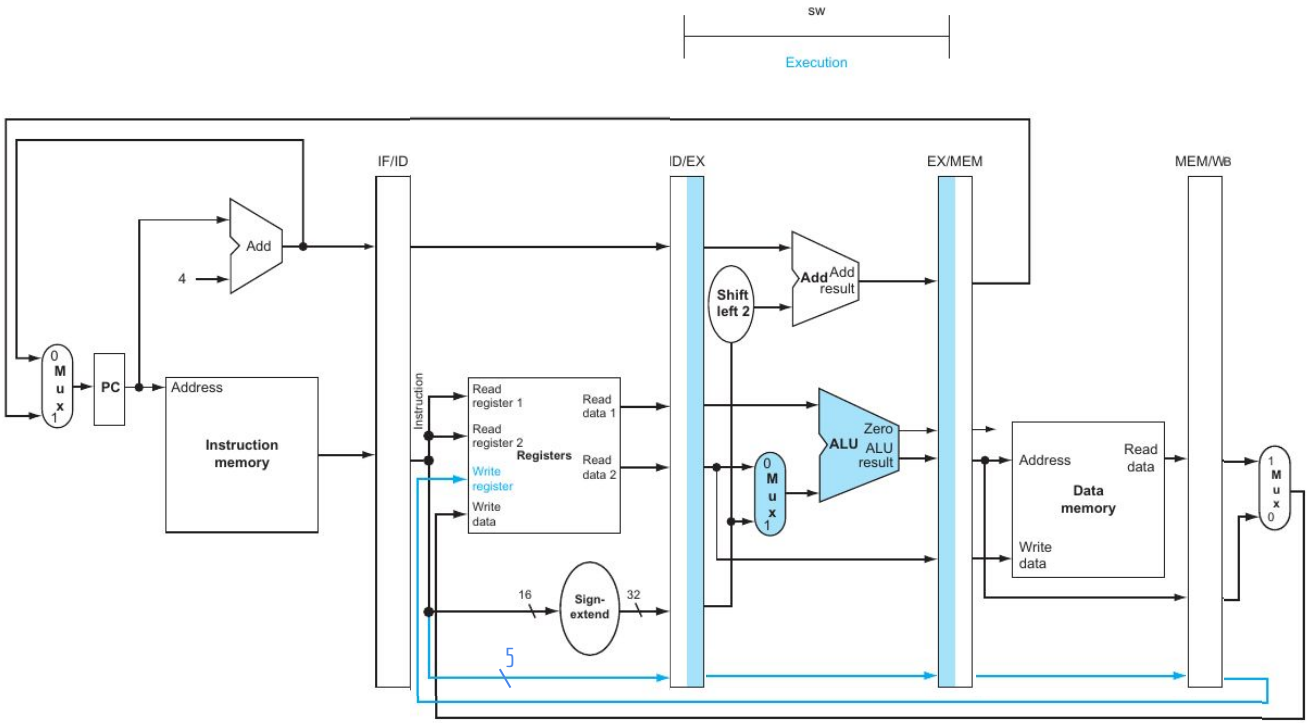


Outro exemplo - sw

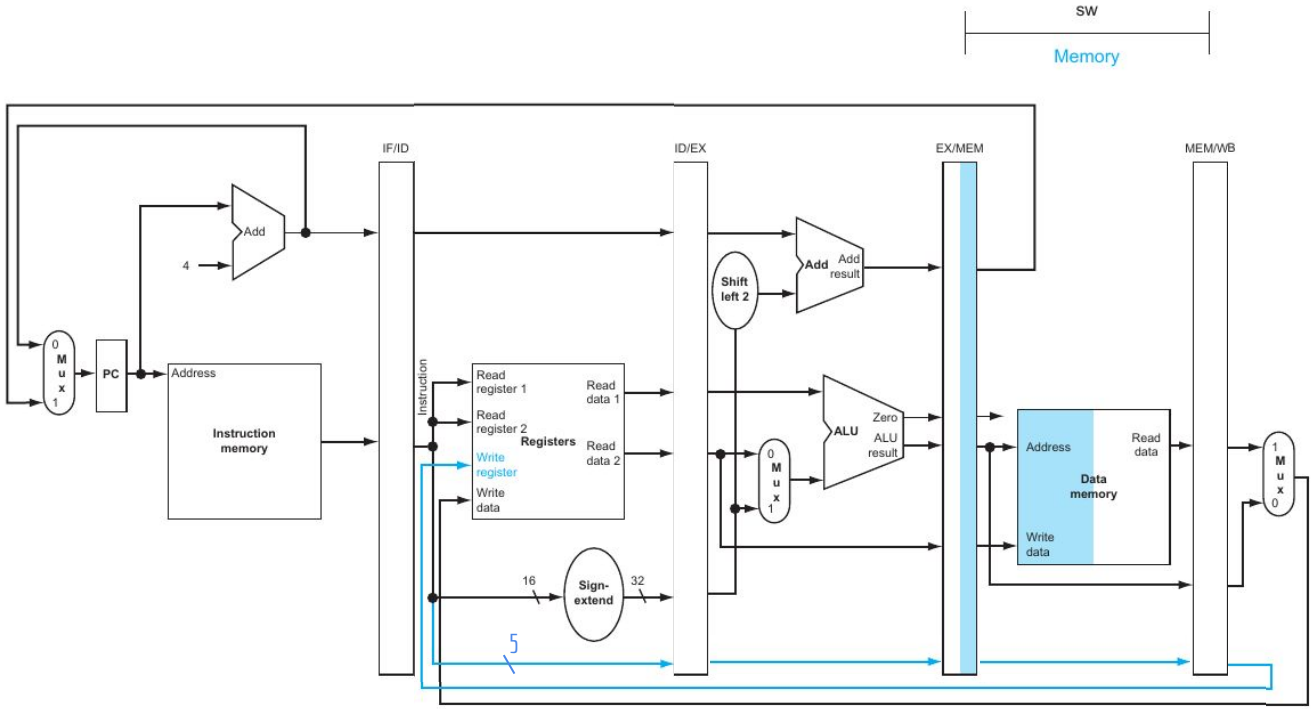
Fluxo de uma instrução sw no pipeline.

Os primeiros estágios do sw são os mesmos do lw.

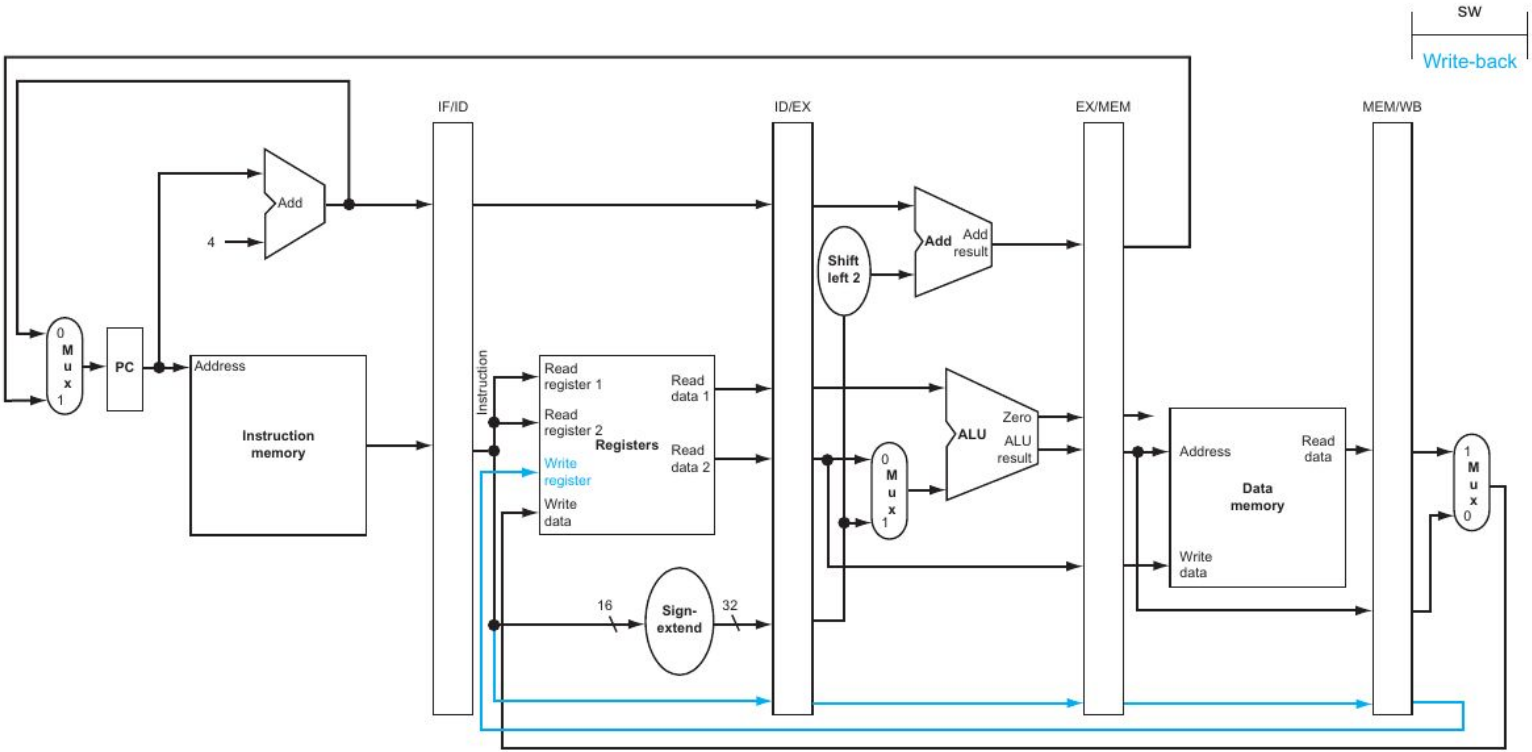
sw no Pipeline



sw no Pipeline



sw no Pipeline



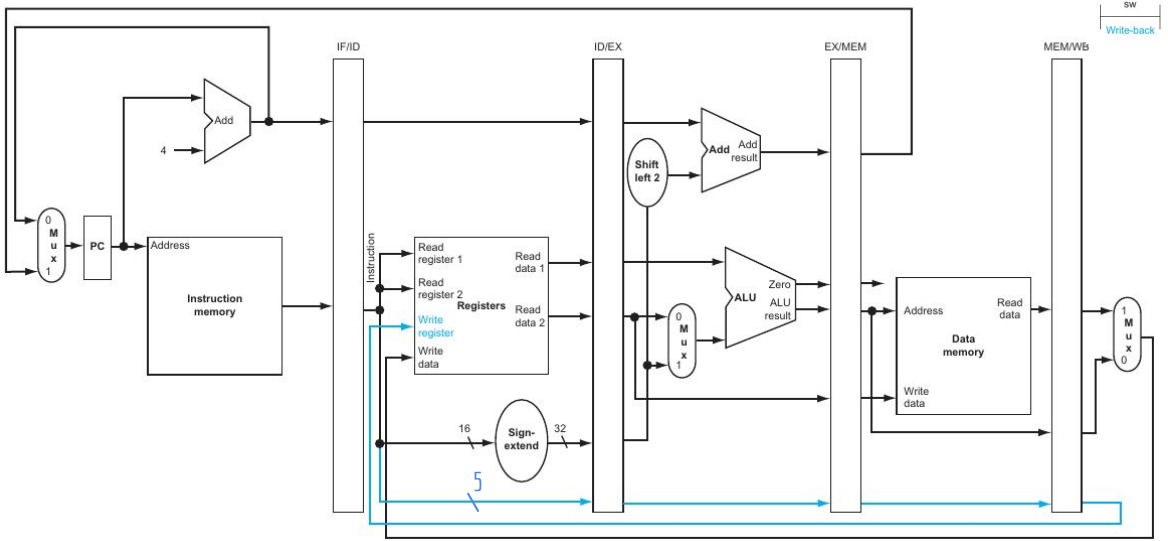
sw no Pipeline

No estágio WB, a instrução sw não realiza trabalho algum.

Ainda assim não podemos “pular estágios”.

Adiantar a execução da instrução que vem logo após o sw não pode ser feito.

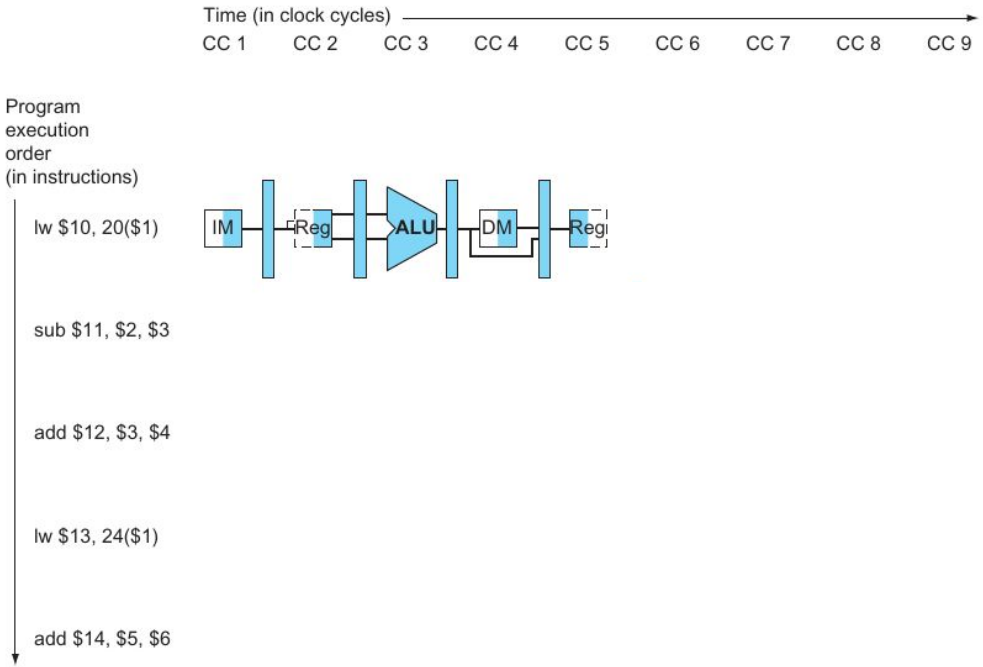
O estágio anterior pode ainda não ter terminado o seu trabalho.



Faça você mesmo

Considere as instruções

```
lw $10,20($1)
sub $11,$2,$3
add $12,$3,$4
lw $13,24($1)
add $14,$5,$6
```



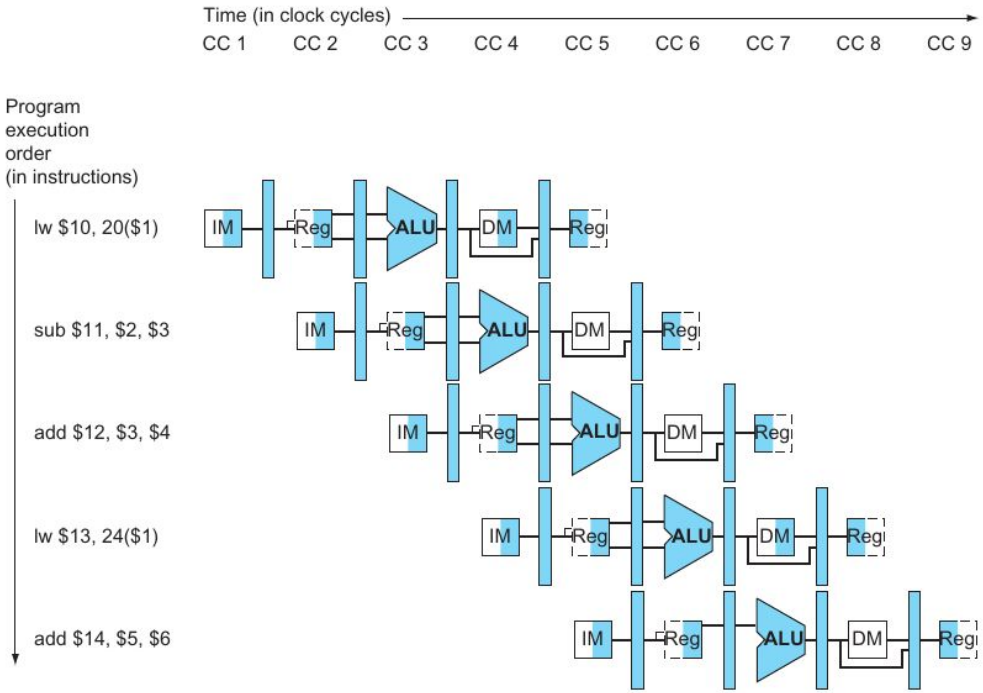
Faça um diagrama de múltiplos ciclos de clock para essas instruções (veja o exemplo para o lw).

Em cada componente do diagrama, pinte-o de acordo com o exemplo para indicar que a unidade está sendo utilizada naquele estágio (ou deixe sem pintar caso não seja usada).

Faça você mesmo

Considere as instruções

```
lw $10,20($1)
sub $11,$2,$3
add $12,$3,$4
lw $13,24($1)
add $14,$5,$6
```



Faça um diagrama de múltiplos ciclos de clock para essas instruções (veja o exemplo para o lw).

Em cada componente do diagrama, pinte-o de acordo com o exemplo para indicar que a unidade está sendo utilizada naquele estágio (ou deixe sem pintar caso não seja usada).

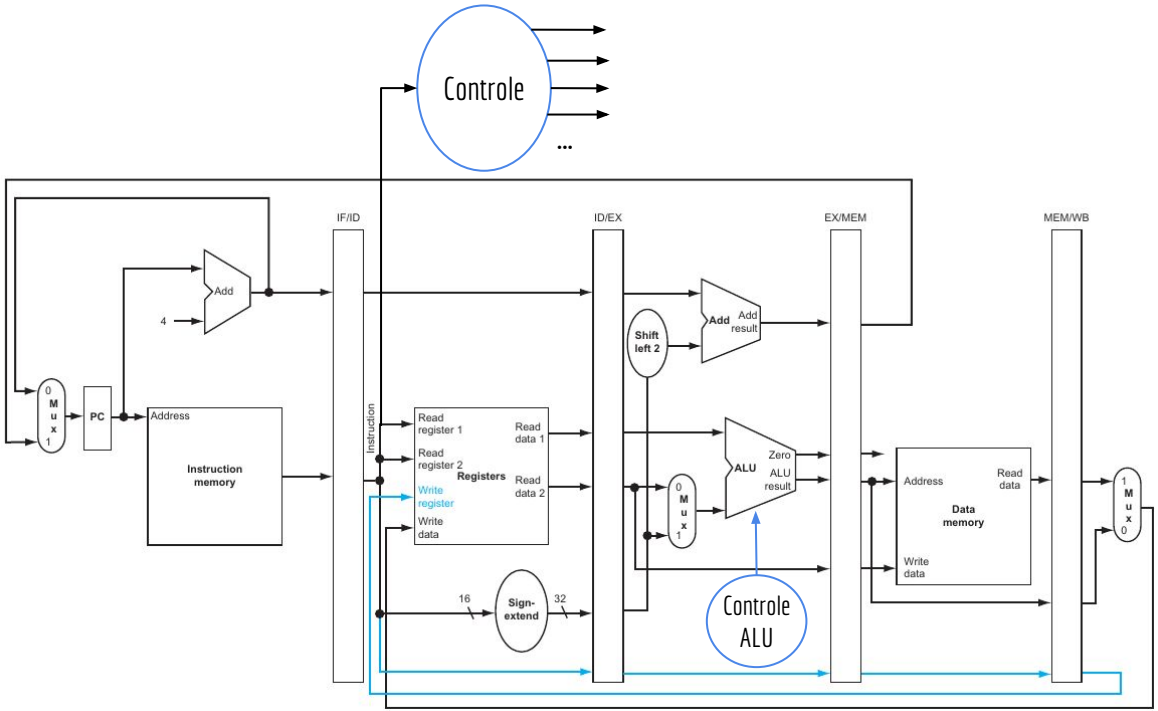
Controle

Os sinais de controle são (por enquanto) os mesmos que na máquina de ciclo único.

O sinal pode ser definido já no estágio ID.

Podemos simplesmente ligar
os sinais na CPU?

Problemas?

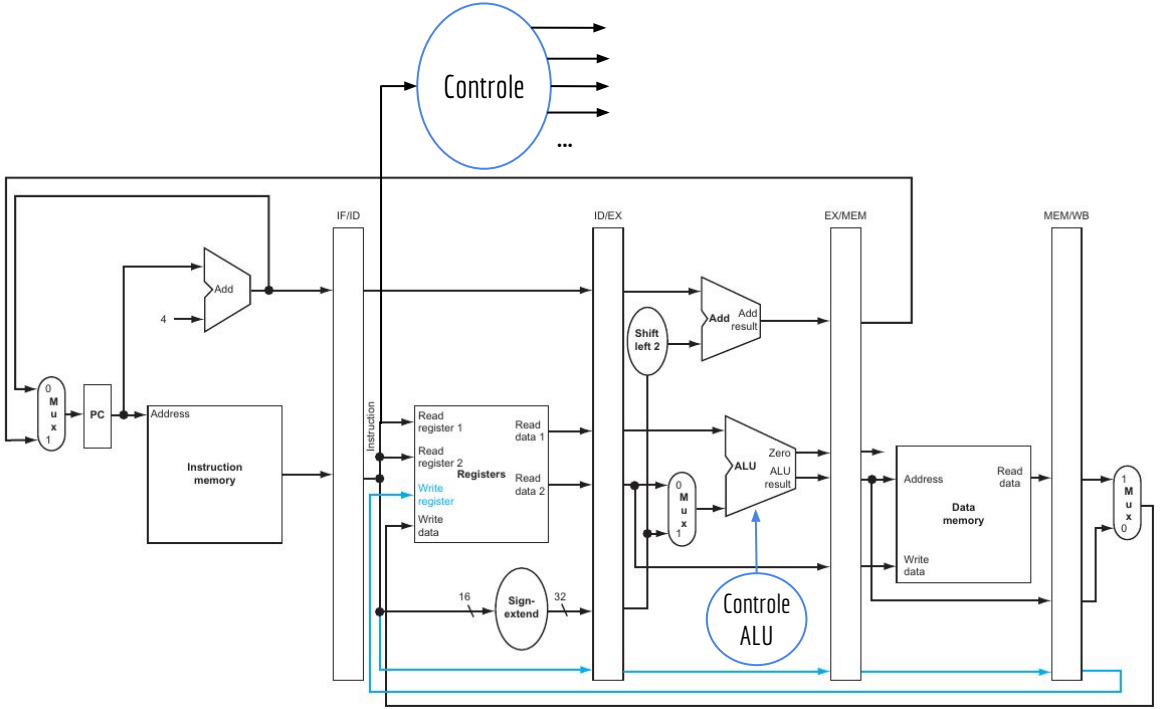


Controle

Diferentes sinais são utilizados em diferentes estágios do nosso pipeline.

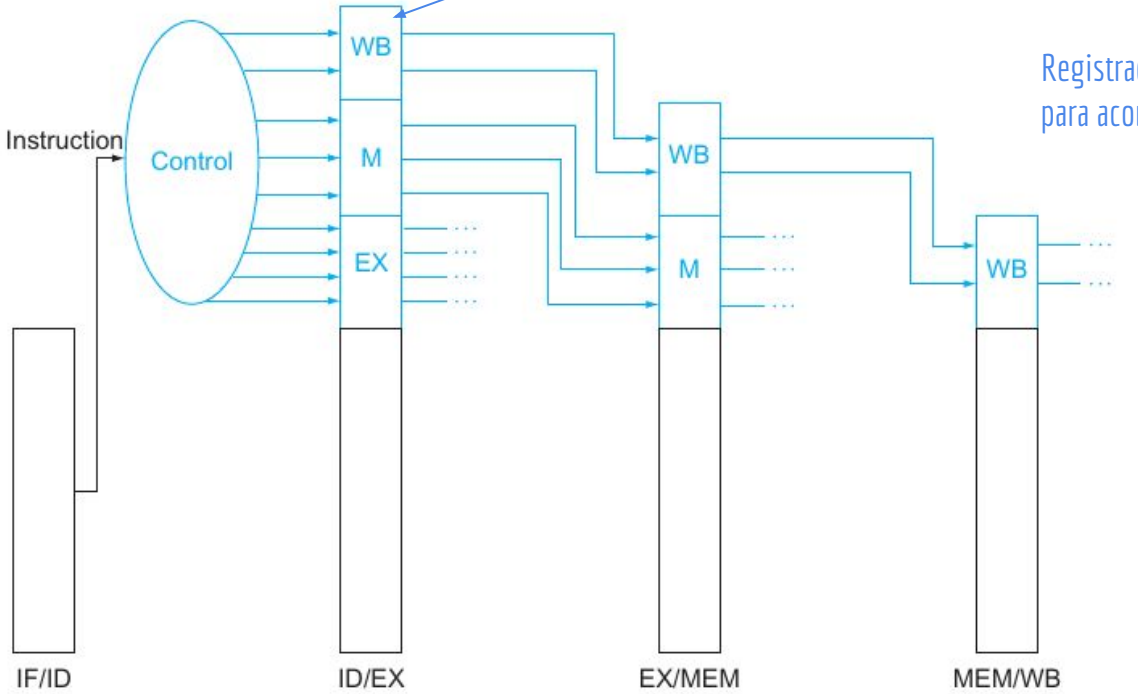
Devemos salvar esses sinais.

Registradores de Pipeline.



Controle

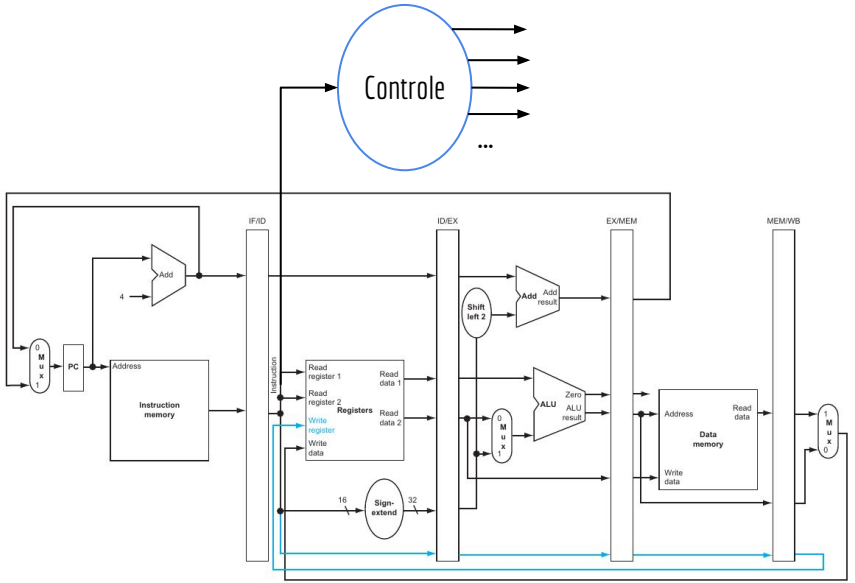
Exemplo: **WB** indica que esses registradores estão salvando os sinais de controle que serão utilizados no estágio WB.



Registradores de pipeline estendidos para acomodar os sinais de controle.

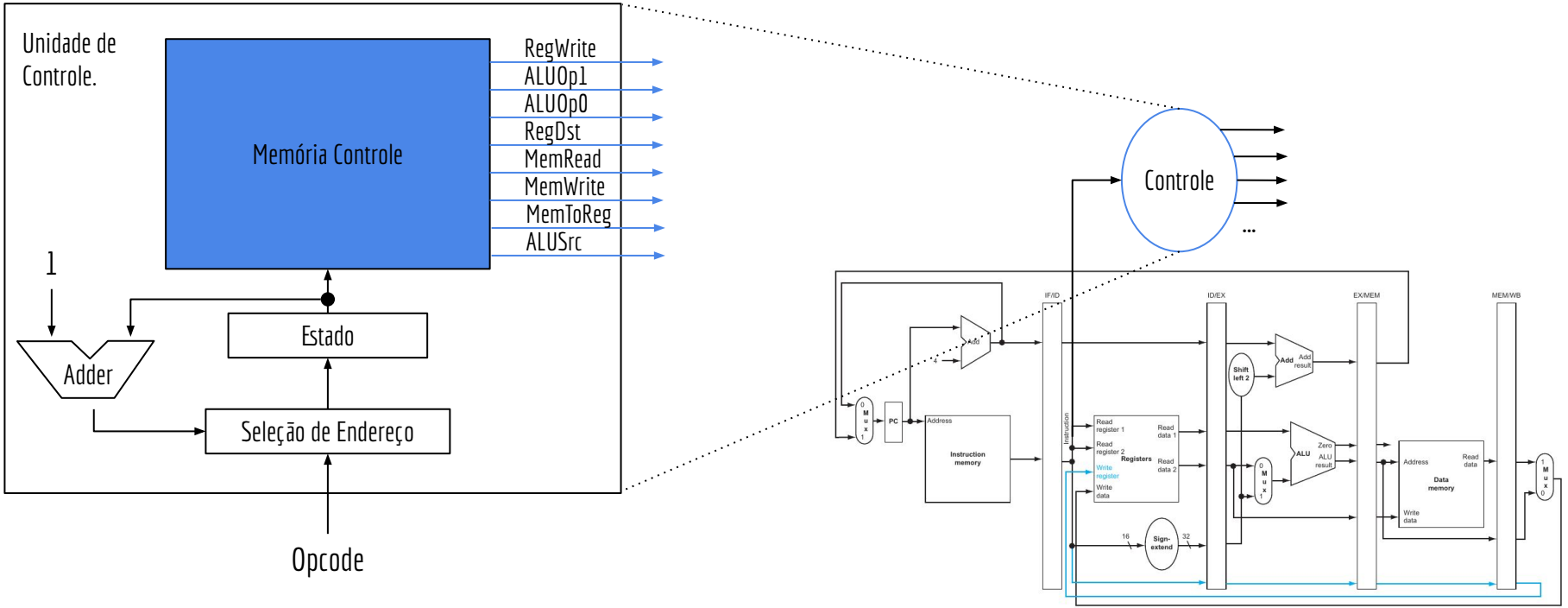
Um pouco mais de microprograma

A unidade de controle pode ser hardwired, como visto na aula sobre a CPU monociclo.

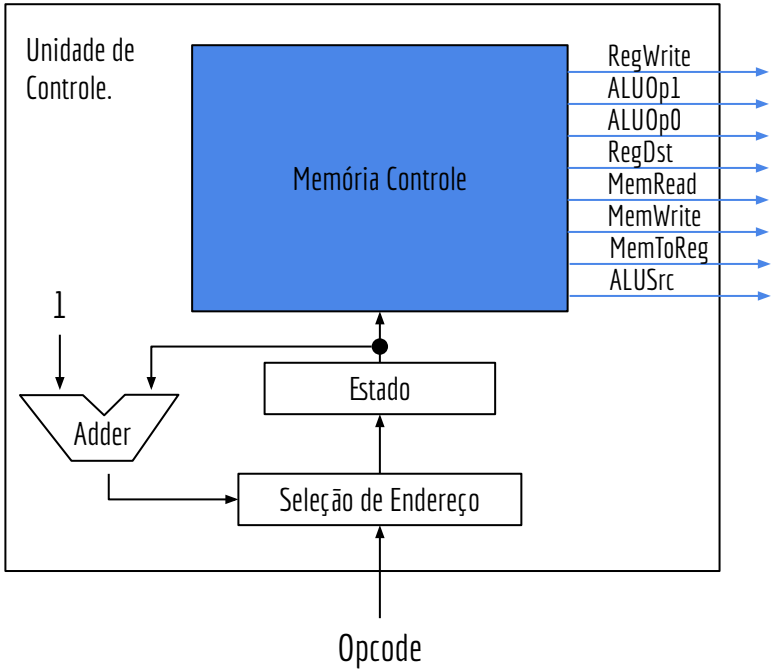


Um pouco mais de microprograma

Ou podemos criar uma “mini CPU” dentro da CPU, que vai gerar as saídas de controle.



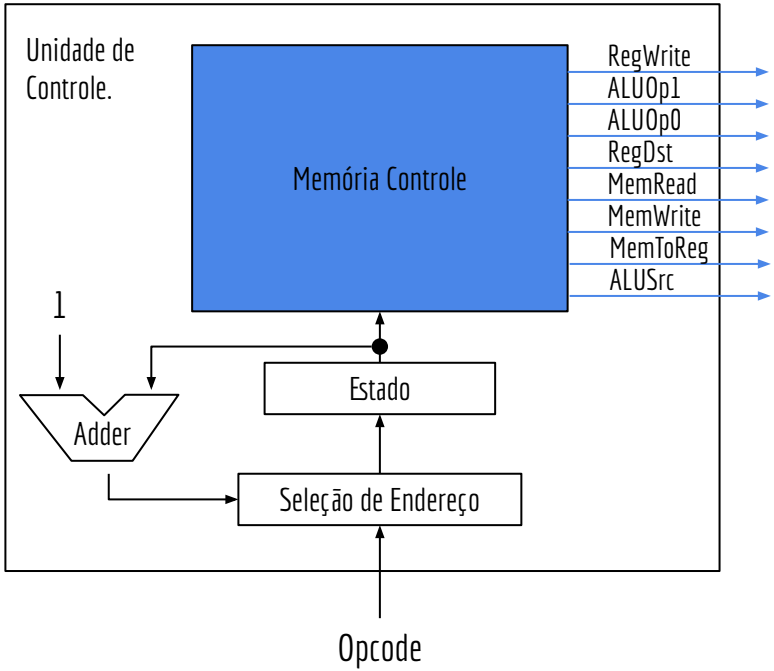
Um pouco mais de microprograma



Podemos criar uma versão do assembly para a unidade de controle com microcódigo.

Cada instrução é chamada de microinstução.

Um pouco mais de microprograma



Podemos criar uma versão do assembly para a unidade de controle com microcódigo.

Cada instrução é chamada de microinstução.

Microoinstrução `00110010LW2`

Um bit para cada sinal de controle.

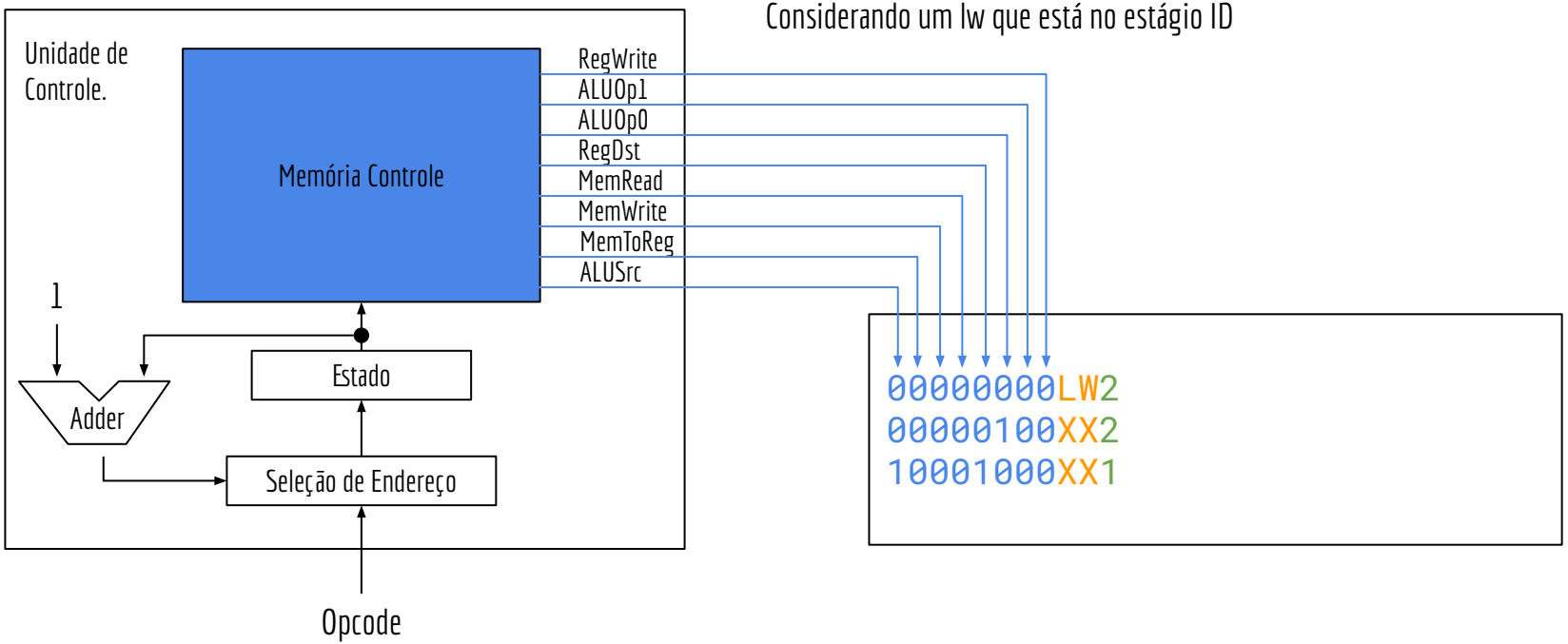
Bits para condição de desvio (no exemplo, se o opcode for LW, desviar).

Endereço de desvio.

Um pouco mais de microprograma

Exemplo de microprograma.

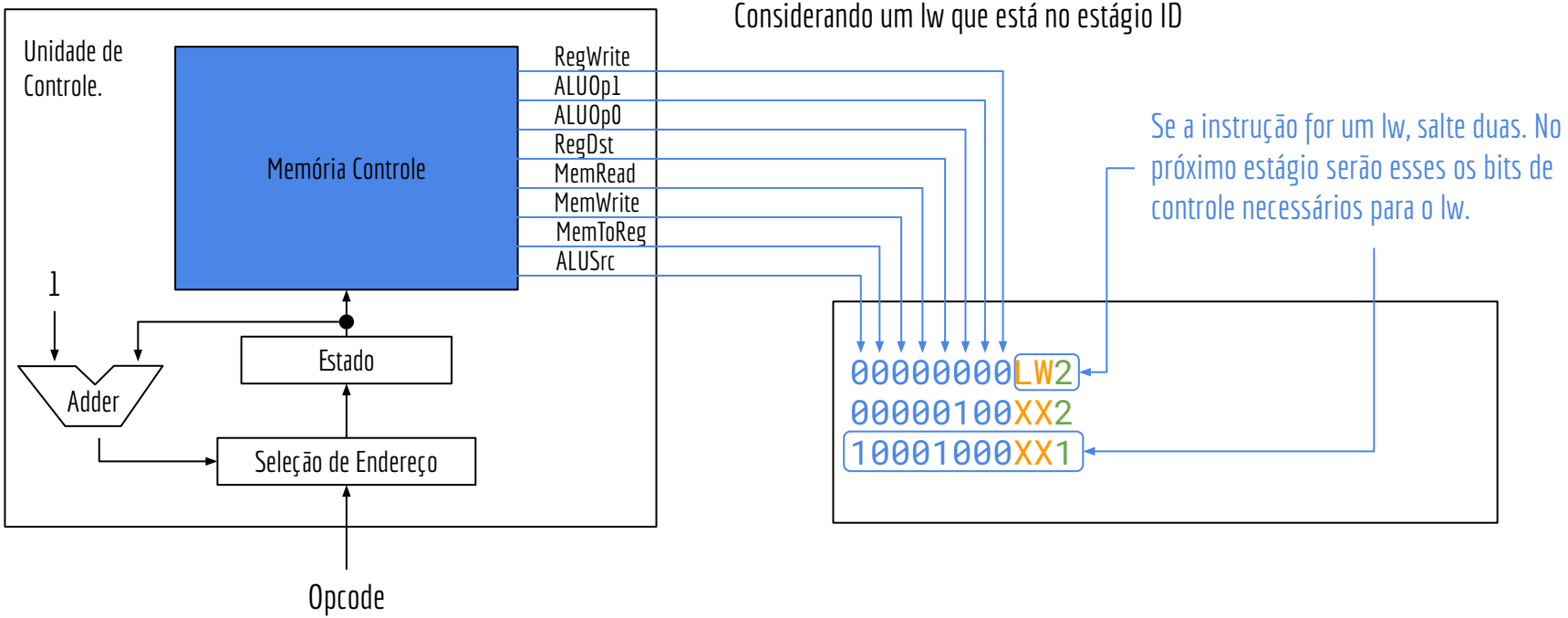
Considerando um lw que está no estágio ID



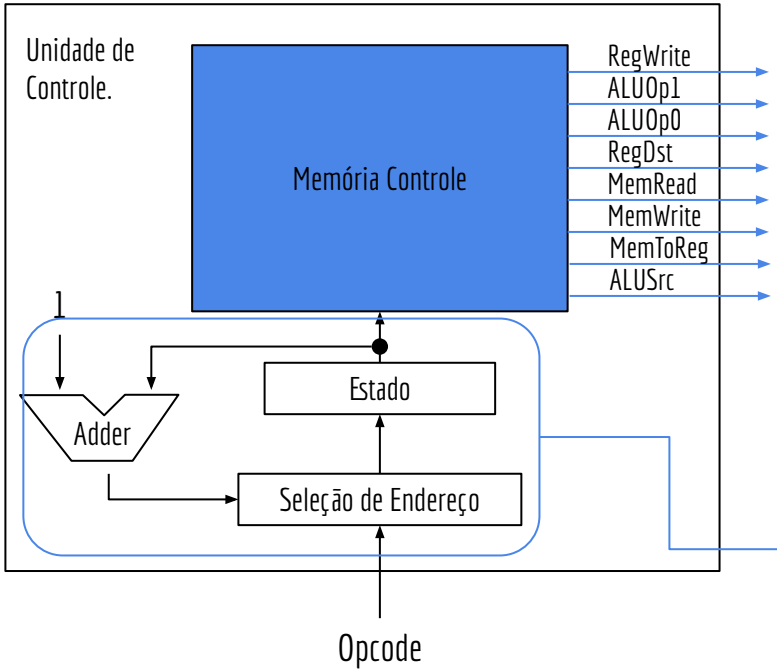
Um pouco mais de microprograma

Exemplo de microprograma.

Considerando um lw que está no estágio ID



Um pouco mais de microprograma



Ter um adder indicando que existem múltiplos ciclos de controle para uma instrução faz sentido em CPUs multiciclo (ou em trechos multiciclo das cpus).

Um pouco mais de microcódigo

Atenção: essa é uma ideia **muito** simplificada do princípio de um microcódigo.

Servem para você vislumbrar como as coisas podem ser feitas.

Muitas complexidades estão ocultas.

Microcódigo faz mais sentido em CPUs **multiciclo**, ou em trechos **multiciclo** de CPUs.

A instrução usa os mesmos blocos funcionais em um “loop” até terminar.

Ex.: primeiro usa a ALU para calcular o resultado, depois usa para computar PC+4.

Veja mais detalhes em:

Patterson, Hennessy. Computer Organization and Design RISC-V Edition. 2020.

Stallings, W. Organização de Arquitetura de Computadores. 10a Ed. 2016.

Um pouco mais de microcódigo

Uma unidade de controle microprogramada é:

- \+ Mais flexível do que uma unidade hardwired.
- \+ Mais simples do que hardwired.
- \+ Pode ser atualizada.

Um pouco mais de microcódigo

Uma unidade de controle microprogramada é:

+ Mais flexível do que uma unidade hardwired.
+ Mais simples do que hardwired.
+ Pode ser atualizada.

- Podem ser mais lentas do que o controle Hardwired.

Um pouco mais de microcódigo

Uma unidade de controle microprogramada é:

+ Mais flexível do que uma unidade hardwired.
+ Mais simples do que hardwired.
+ Pode ser atualizada.

- Podem ser mais lentas do que o controle Hardwired.

Muitos processadores, como o seu x86-64, usam uma combinação de controle hardwired e microprogramado.

Hardwired para instruções simples, e microprogramado para instruções complexas.

Armazenamento

O microcódigo geralmente é armazenado em uma memória ROM, ou criado em um Arranjo Lógico Programável.

Algumas CPUs podem ainda usar memórias SRAM ou Flash.

Possibilitar a atualização do microprograma.

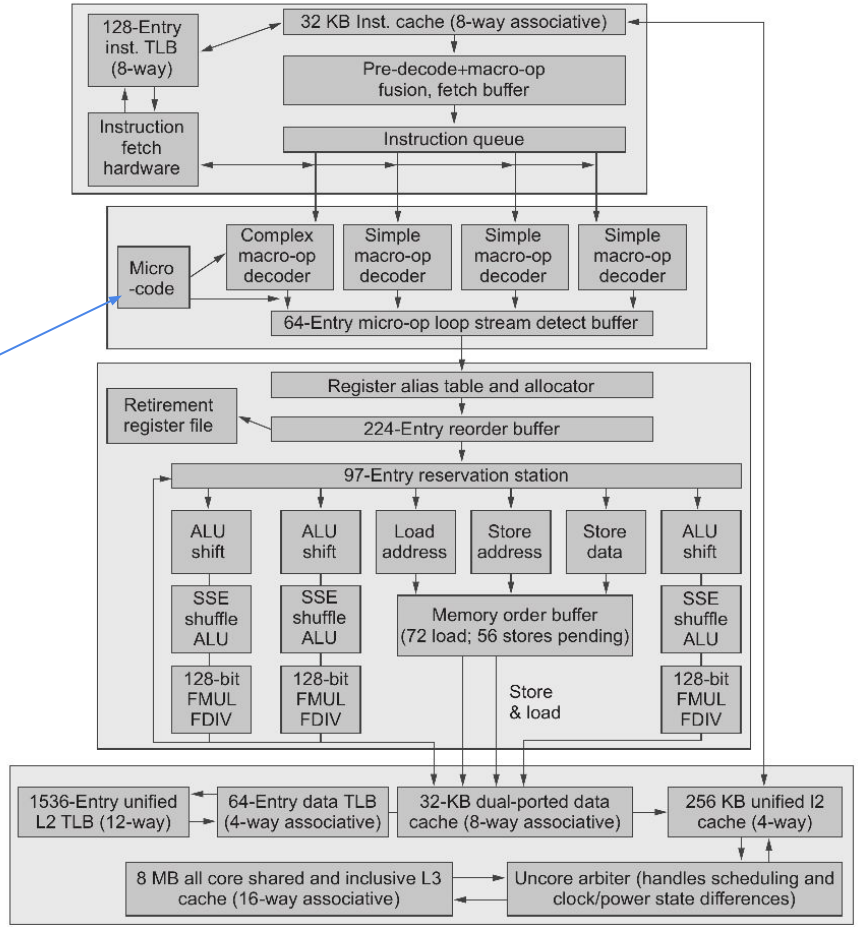
Seu x86-64 por exemplo, possibilita que uma versão atualizada do microcódigo seja carregada na sequência de boot pelo Sistema Operacional.

www.kernel.org/doc/html/latest/x86/microcode.html

Exemplo

Estrutura de um Intel i7.

Microcódigo para “decifrar” instruções complexas.



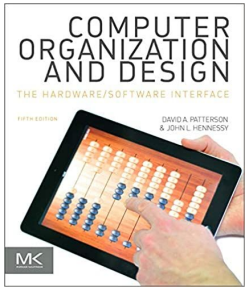
Patterson, Henessy, 2020.

Exercícios

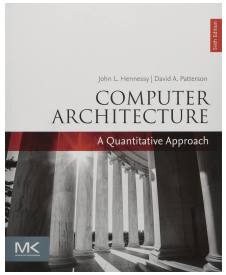
1. Adicione a unidade de controle e ligue os sinais. Estenda os registradores de pipeline onde for necessário (Resposta no livro base da disciplina).
2. Considere os registradores de pipeline do exercício anterior, agora com os sinais de controle. Qual o tamanho de cada um dos registradores de pipeline (IF/ID, ID/EX, ...)?

Referências

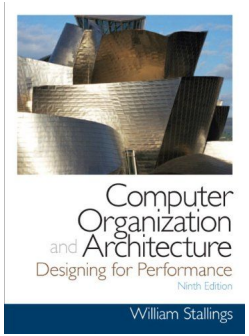
Patterson, Hennessy . Arquitetura e Organização de Computadores: A interface hardware/software. 2014.



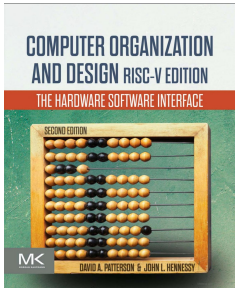
Hennessy, Patterson. Arquitetura de Computadores: uma abordagem quantitativa. 2019.



Stallings, W. Organização de Arquitetura de Computadores. 10a Ed. 2016.



Patterson, Hennessy. Computer Organization and Design RISC-V Edition: The Hardware Software Interface. 2020.



Licença