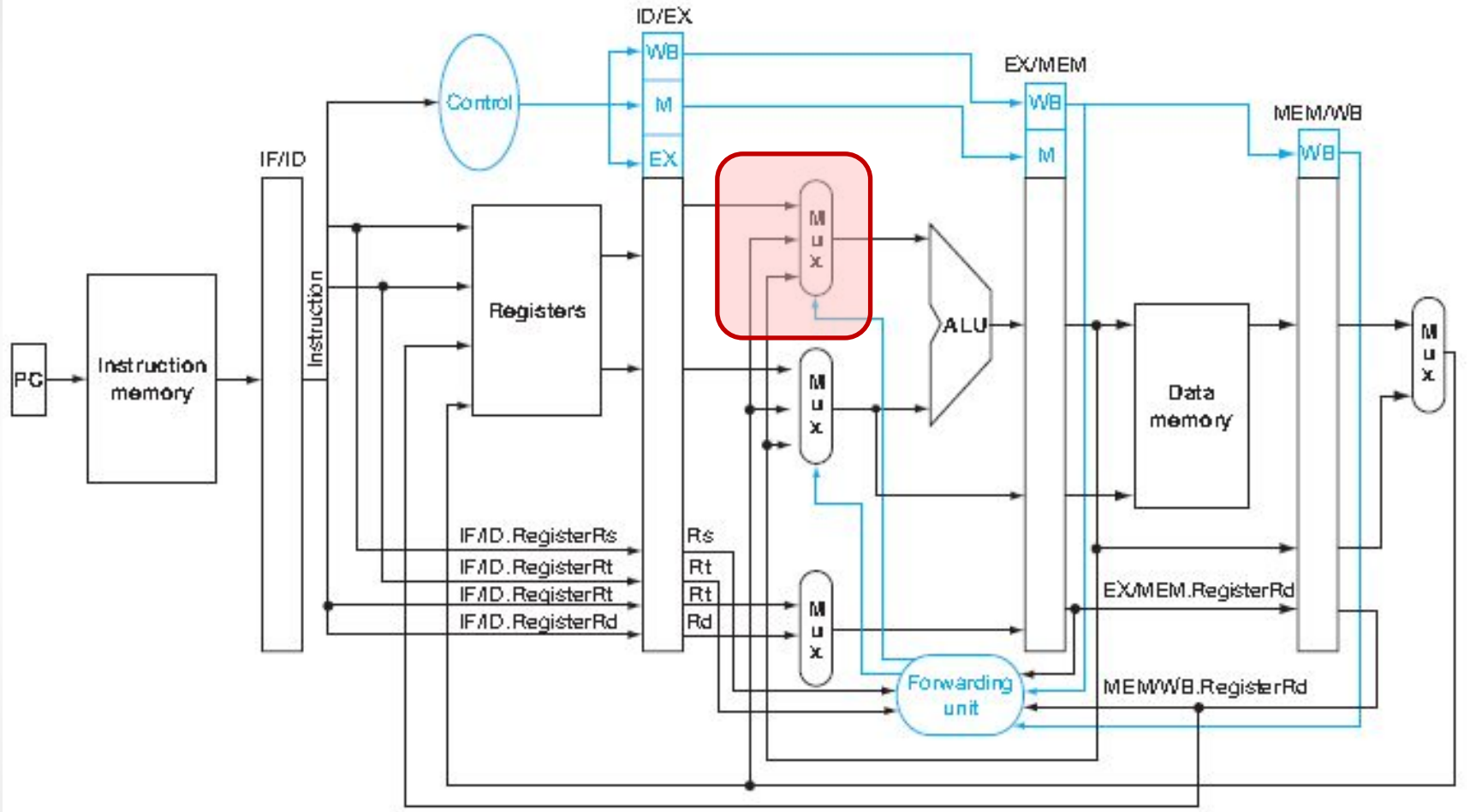
Aula 6

Organização de Computadores

Pipeline e Paralelismo a nível de instruções

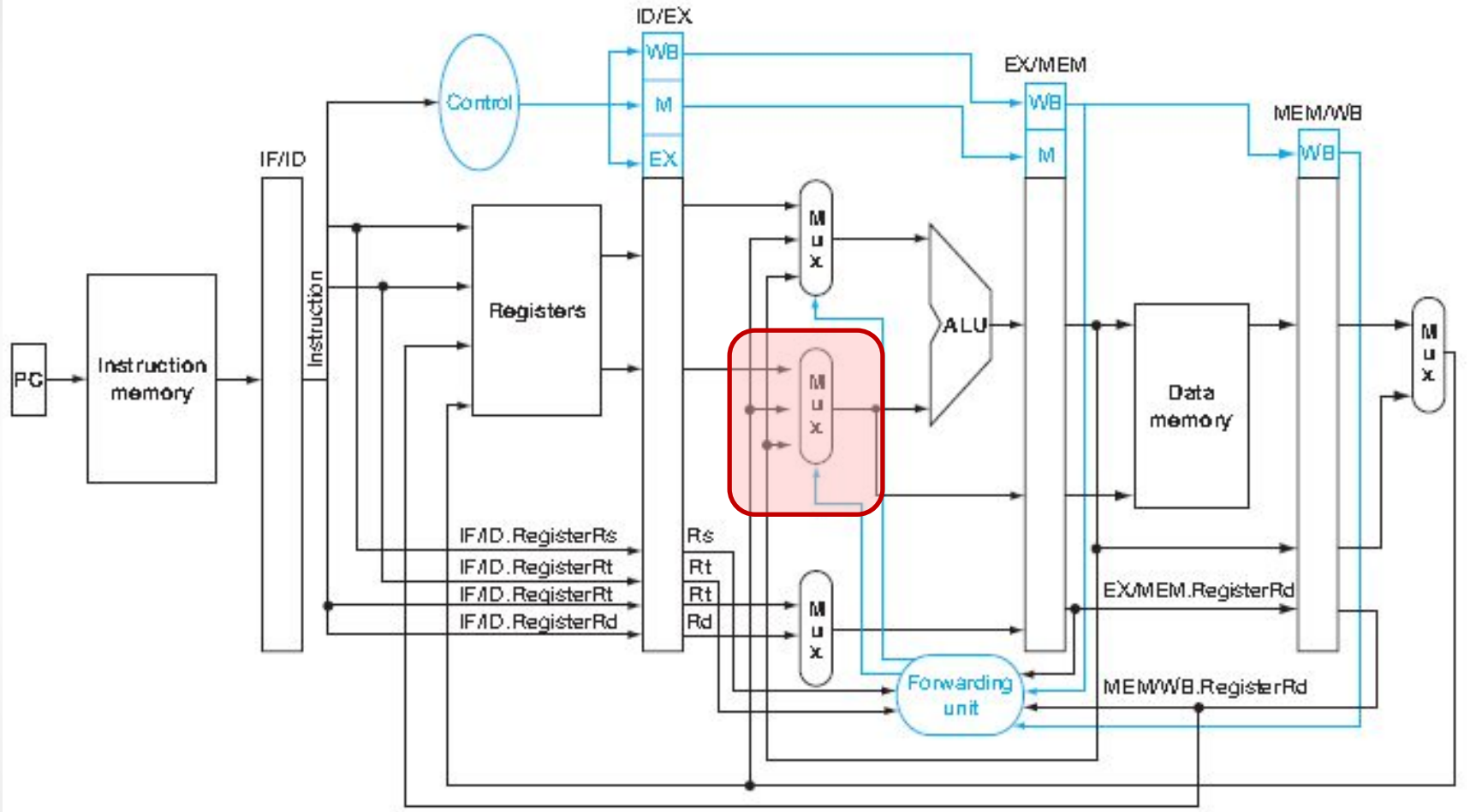
Profa. Débora Matos

Hazards de dados: implementação de forwarding



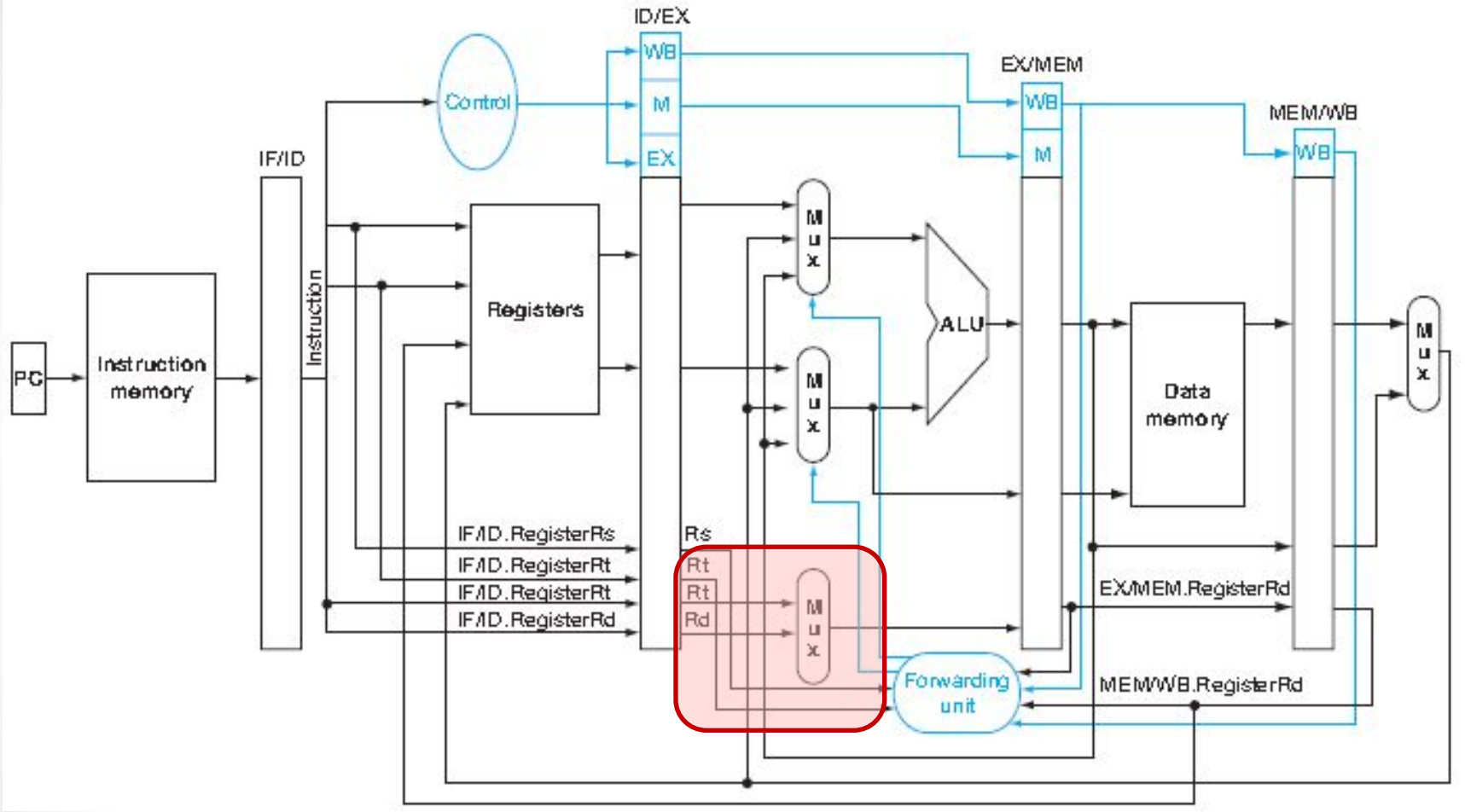
- Primeiro operando da ALU:
  - Banco de registradores
  - Forwarding da saída da memória (exemplo: lw – add)
  - Forwarding da saída da ALU (exemplo: add – add)

Hazards de dados: implementação de forwarding



- Segundo operando da ALU:
  - Banco de registradores
  - Forwarding da saída da memória (exemplo: lw – add)
  - Forwarding da saída da ALU (exemplo: add – add)

Hazards de dados: implementação de forwarding



- Definição do registrador de destino:
  - Rt – para instruções com 2 registradores (instrução do tipo I)
  - Rd – para instruções com 3 registradores (instruções do tipo R)

Hazards de dados: implementação de forwarding

- Além da implementação do forwarding, é necessário acrescentar uma unidade de detecção de hazard.
- A unidade de controle de hazard opera durante o estágio de ID.
  - Exemplo 1: add $s2, $s3, $s1
    and $s4, $s2, $s0
- 1° testa se a instrução anterior e subsequente é do tipo-R
- 2° verifica se o campo do registrador de destino refere-se a um dos registradores de origem da instrução subsequente.
- Se sim, seleciona a opção do mux para realizar o forwarding.

Hazards de dados: implementação de stall

- Exemplo 2: lw $s2,20($s1)
  and $s4, $s2, $s0
- Nesse caso é necessário inserir primeiramente um ***stall.***
  1. testa se a instrução é load
  2. verifica se o campo do registrador de destino do *load* refere-se a um dos registradores de origem da instrução subsequente.
  3. Se sim, é necessário acrescentar um ***stall*** de um ciclo no pipeline.
  4. A seguir, é aplicado o *forwarding.*

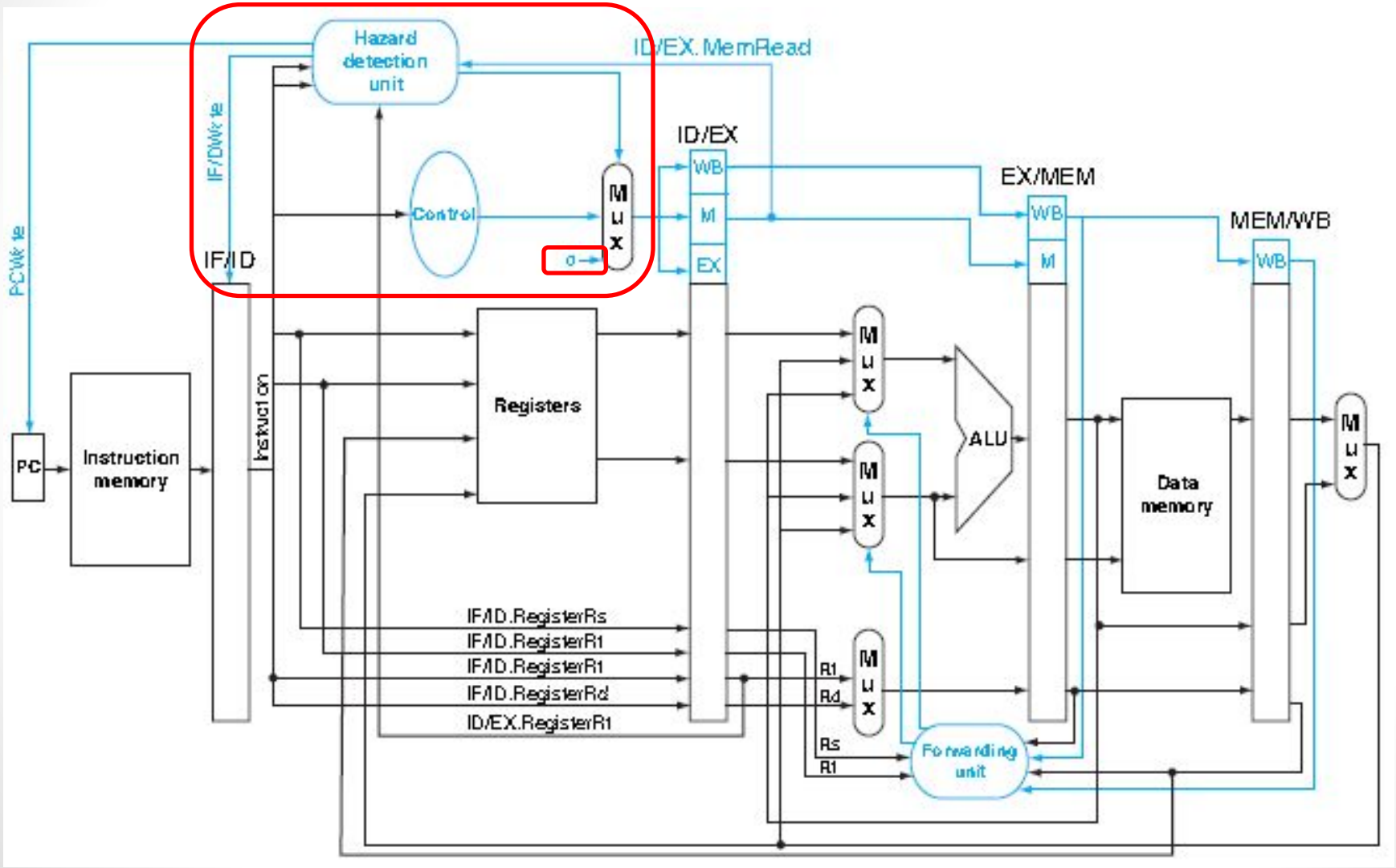
Hazards de dados: implementação de stall

- O que significa acrescentar *stall*?
  - Uma possível solução é colocar as linhas de controle em 0 (zero), de modo que nada seja alterado e feito;
  - Mantendo os controles em 0 (zero), nenhum registrador ou memória serão alterados. De fato, manter apenas os sinais ***EscreveReg* e *EscreveMem* em 0 (zero)** já garante que nada seja alterado. Assim, os demais sinais podem ser "*don't care*";

    Qual outro controle precisa ser considerado??

  - O PC também não deve ser incrementado;
  - Como fazer esse controle?

Hazards de dados: implementação de stall



Hazards de controle: desvio não é tomado

- É apostado que o desvio não é tomado. Nesse caso, quando o desvio for tomado, as instruções que entraram no pipeline precisam ser descartadas.
- Descartar instruções é semelhante a produzir *stall*, precisa-se garantir que nada vai ocorrer;
- Nesse caso, a diferença é que as instruções, até que a decisão seja obtida, precisam ser descartadas;
- As 3 instruções nos estágios IF, ID e EX precisam ser alteradas quando a decisão de desvio não for acertada.
- Simplesmente o controle é alterado para 0 no estágio ID;
- Descartar instruções significa dar flush nas instruções IF, ID e EX.

Hazards de controle: Antecipando o desvio

Reduzindo o atraso dos desvios:

- Os desvios contam com testes simples que não exigem uma operação completa da ALU, podendo ser feitos com apenas algumas portas lógicas;
- É possível mover a execução do desvio para um estágio anterior do pipeline. Isso requer que 2 ações ocorram mais cedo. Quais?
  1. Calcular o endereço de destino do desvio
  2. Avaliar a decisão do desvio

Hazards de controle: Antecipando o desvio

1. Calcular o endereço de destino do desvio:

- O somador dos desvios é acrescentado no estágio anterior a execução da ULA. O cálculo é sempre realizado, mas só é utilizado quando o desvio deve ser tomado.

2. Avaliar a decisão de desvio:

- Para *branch equal*, a igualdade pode ser comparada durante o estágio ID (leitura dos registradores). Na comparação, pode ser utilizado uma XOR e depois uma OR de todos os bits.

Hazards de controle: Antecipando o desvio

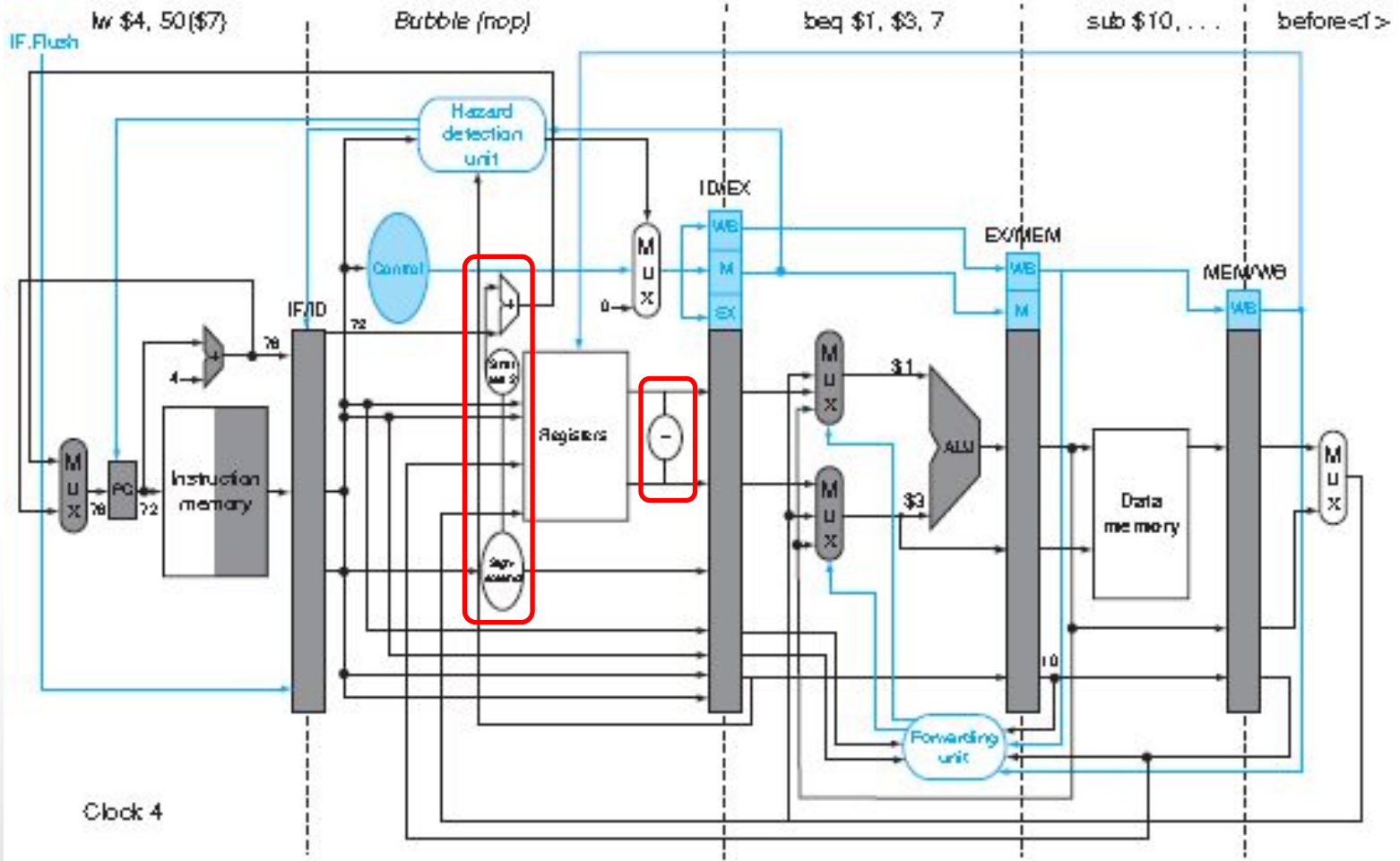
- Com a decisão de adiantar o desvio, é possível que ocorra um *hazard* de dados e um *stall.* Quais são estas situações??

- Por exemplo, se estiver sendo executada uma instrução da ALU imediatamente antes do desvio que produz um dos operandos para a comparação, um stall será exigido;

- Se uma instrução de lw antevir o desvio condicional e o resultado do load precisa ser usado na comparação, 2 ciclos de *stall* são necessários, pois o resultado do *load* aparece somente no final do ciclo de MEM (4° ciclo);

Hazards de controle: Antecipando o desvio

- Apesar das considerações, mover a instrução de desvio para o estágio ID é uma melhoria, reduzindo a penalidade de 1 desvio a apenas 1 instrução, por exemplo, se o desvio for tomado.



Hazards de controle: Previsão Dinâmica

- **Previsão dinâmica de desvios**: pesquisa o endereço da instrução para verificar se o desvio foi tomado na última vez que a mesma foi executada.
- Uma implementação desta técnica é obtida a partir de um **buffer de previsão de desvio ou tabela de histórico de desvios.**
  - Trata-se de uma pequena memória indexada pela parte menos significativa do endereço da instrução de desvio.
  - A memória contém um bit que diz se o desvio foi tomado ou não.

Apresente todas as situações que envolvem a antecipação do desvio:

Exceções

- Controle é o aspecto mais desafiador do projeto do processador.
- Ainda no controle, uma das partes mais difíceis de se implementar são as exceções e as interrupções – situações que mudam o fluxo normal de execução das instruções.
- Exceções no MIPS referem-se a qualquer mudança inesperada no fluxo de controle.
- Interrupção no MIPS refere-se a eventos externos.

Tipo de Evento	Terminologia MIPS
Solicitação de dispositivo de E/S	Interrupção
Chamada do sistema operacional por um programa de usuário	Exceção
Overflow aritmético	Exceção
Usar uma instrução indefinida	Exceção
Defeitos do hardware	Exceção ou interrupção

Exceções

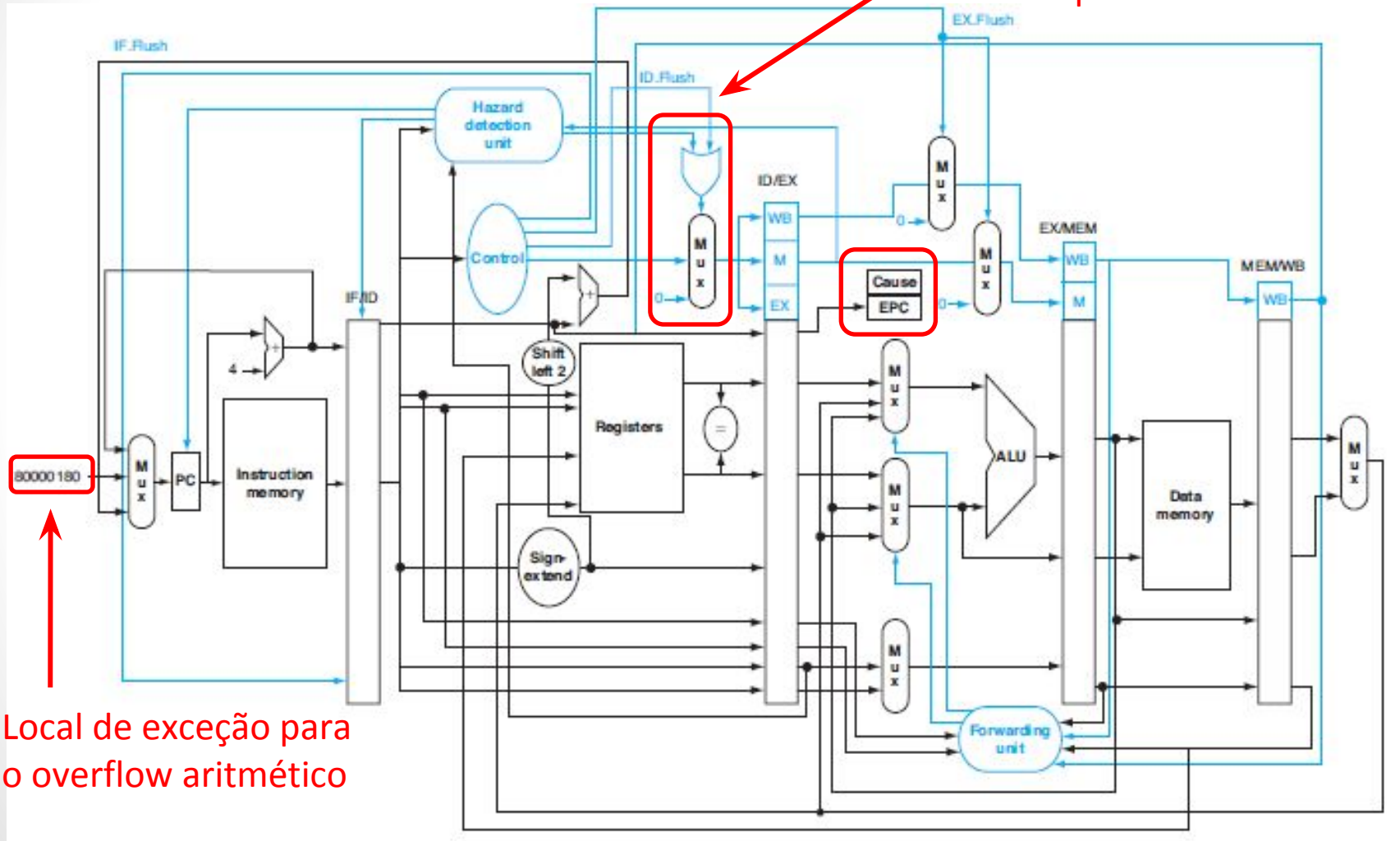
- Vamos analisar 2 tipos de exceções:
  - ✓ Instruções indefinidas
  - ✓ Overflow aritmético
- Quando ocorrer uma exceção, o processador deve salvar o endereço da instrução no PC de execução (**EPC – Exception Program Counter**) e depois transferir o controle para o sistema operacional;
- Algumas possíveis ações:
  - ✓ Tomar uma ação predefinida
  - ✓ Terminar a execução do programa e informar um erro
- Após a ação à exceção, o sistema operacional pode decidir terminar o programa ou continuar sua execução, usando o EPC para determinar de onde reiniciar a execução do programa.

Exceções

- Para o sistema operacional tratar a exceção, ele precisa identificá-la. O MIPS possui um registrador de status (registrador Cause) para indicar o motivo da exceção.
- Na organização do MIPS é necessário acrescentar 2 registradores:
  - ✓ **EPC**: registrador de 32 bits usado para manter o endereço da instrução afetada;
  - ✓ **Cause:** registrador que indica a causa da exceção
- No caso de overflow aritmético, é preciso detectar esses casos antes que o valor “errado” seja escrito no registrador de destino.

Exceções

É dado flush nas instruções seguintes, zerando os sinais de controle para ocorrer *stalls*



Local de exceção para o overflow aritmético

A ALU envia um sinal indicando a ocorrência de overflow para a unidade de controle.

Exceções

Interface hardware/software:

- O Hardware e o Sistema Operacional precisam trabalhar em conjunto para que as exceções sejam tratadas.

Função do Hardware:

- Interromper a instrução problemática;
- Deixar que as instruções anteriores terminem;
- Dar *flush* nas instruções seguintes;
- Definir um registrador para mostrar a causa da exceção;
- Salvar o endereço da instrução problemática;
- Desviar para o endereço previamente definido;

Exceções

Função do Sistema Operacional:

- Instruções indefinidas, falha de hardware ou exceção por overflow aritmético, o sistema operacional normalmente encerra o programa e retorna um indicador do motivo;
- Para uma solicitação de um dispositivo de E/S é função do SO salvar e restaurar o estado de qualquer tarefa após o atendimento da interrupção, permitindo o retorno da execução do programa.

Introdução ao paralelismo em nível de instrução

- A técnica de pipeline explora o paralelismo entre as instruções chamado de **paralelismo em nível de instrução (ILP – Instruction Level Parallelism).**
- **Existem 2 métodos principais:**
  1. Aumenta a profundidade do pipeline para sobrepor mais instruções;
     - Ciclo de clock pode ser reduzido, aumentando a frequência de operação.
  2. Despacho múltiplo: Replica os componentes internos do computador de modo que várias instruções possam ser iniciadas em cada estágio do pipeline.

Introdução ao paralelismo em nível de instrução

- No entanto, tanto para *pipelines* profundos como para despachos múltiplos, ocorrem ainda mais problemas de ***hazards*** e restrições quanto ao tipo de instruções que podem ser executadas simultaneamente;
- Funções do pipeline de despacho múltiplo:
  1. Empacotar as instruções em slots de despacho. Esse procedimento é tratado parcialmente pelo compilador;
  2. Lidar com *hazard* de dados e de controle. Em processadores de despacho estático, quase todas as consequências dos hazards de dados e controle são tratadas estaticamente pelo compilador.

Introdução ao paralelismo em nível de instrução

- Como o processador de despacho múltiplo estático normalmente restringe o mix de instruções que podem ser iniciadas em um determinado ciclo de clock, é útil pensar no pacote de despacho como uma única instrução;

- Essa estratégia definiu uma outra técnica: VLIW (Very Long Instruction Word – palavra de instrução muito longa).

- As responsabilidades do compilador VLIW podem incluir previsão estática de desvios e escalonamento de código para reduzir ou impedir hazards.

Introdução ao paralelismo em nível de instrução

- Exemplo de despacho múltiplo estático considerando o MIPS:
  - Despacho de 2 instruções por ciclo
  - Uma instrução de acesso à memória e a outra com operação na ULA
  - Se uma instrução do par não puder ser executada, ela é substituída por um *nop*

Instruction type	Pipe stages							
ALU or branch instruction	IF	ID	EX	MEM	WB			
Load or store instruction	IF	ID	EX	MEM	WB			
ALU or branch instruction		IF	ID	EX	MEM	WB		
Load or store instruction		IF	ID	EX	MEM	WB		
ALU or branch instruction			IF	ID	EX	MEM	WB	
Load or store instruction			IF	ID	EX	MEM	WB	
ALU or branch instruction				IF	ID	EX	MEM	WB
Load or store instruction				IF	ID	EX	MEM	WB

Introdução ao paralelismo em nível de instrução

- **Implementação de despacho duplo no MIPS:**