

Organização de Computadores

Aulas 14

Superescalaridade

Janela de Instruções Distribuídas
Renomeação de Registradores
Reordenamento

Superescalaridade – aula de hoje

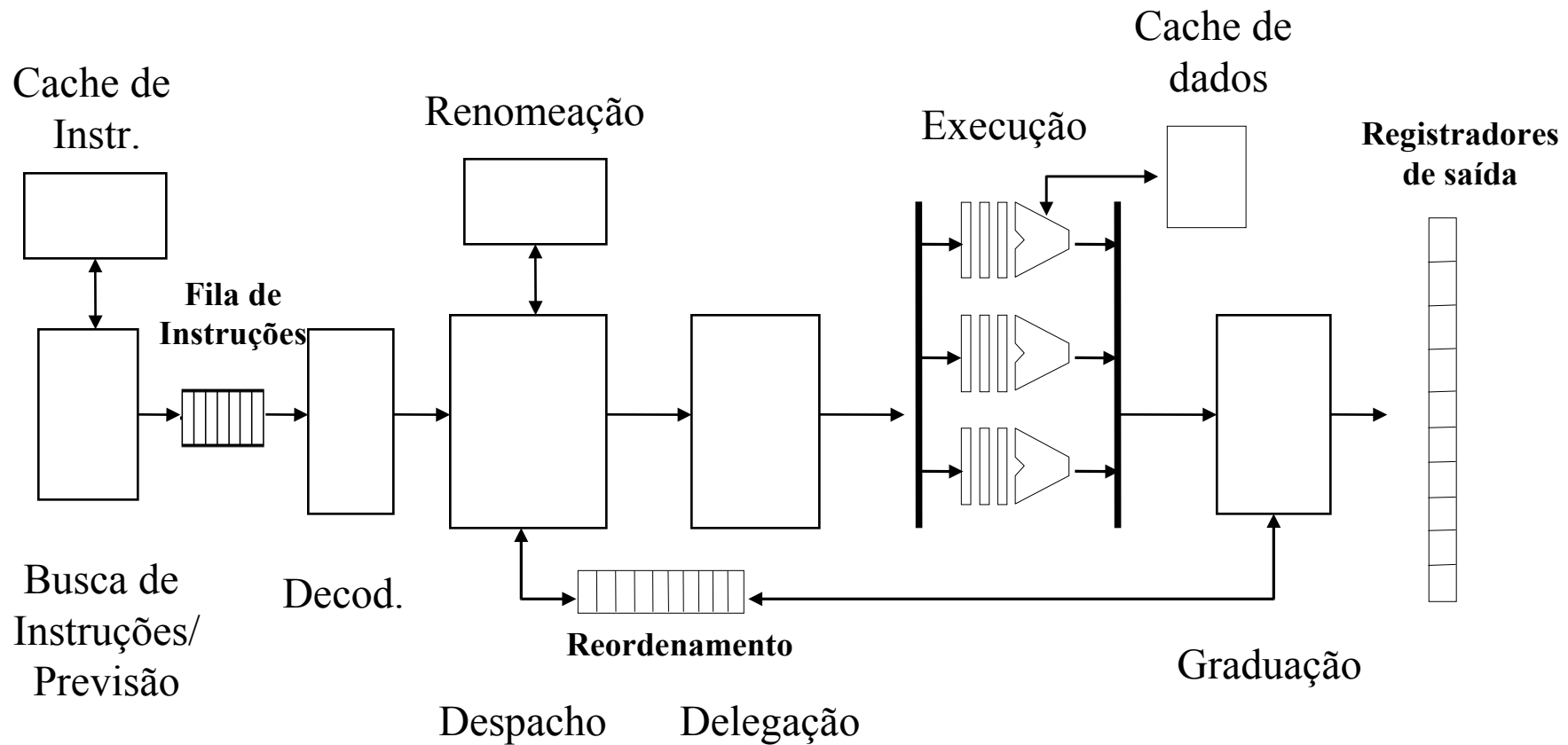
6. Janela de Instruções Distribuída

7. Exemplo

8. Renomeação de Registradores

9. Buffer de Reordenamento

Arquitetura Superescalar



Superescalaridade - Aula Passada

1.Introdução

2.Arquiteturas Superescalares

3.Despacho em ordem, Finalização em ordem

4.Despacho em ordem, Finalização fora-de-ordem

5.Despacho fora-de-ordem, Finalização fora-de-ordem

6.Janela de Instruções Centralizada

Revisão de dependências de dados

Dependências de dados

Verdadeiras

- **Exemplo:**
 - **sub R2, R4, R5 : R5 = R2 - R4;**
 - **add R7, R5, R10 : R10 = R7 + R5;**
- **Instrução 2 depende do valor de ‘R5’, calculado na instrução 1**
- **Valor de ‘R5’ precisa ser gravado antes que a instrução 2 busque seus operandos**
- **Pipeline precisa ser parado durante alguns ciclos**
- **Dependência também conhecida como RAW (read-after-write)**

Dependências de dados

Falsas

- **Em arquiteturas Superescalares, instruções podem ser executadas fora de ordem. Surgem as dependências falsas**
- **Antidependência**
 - **Exemplo**
 - **add R4, R6, R7** : **$R7 = R4 + R6;$**
 - **sub R1, R2, R4** : **$R4 = R1 - R2;$**
 - **Instrução 1 utiliza operando (R4) que é escrito pela instrução 2**
 - **WAR (write-after-read)**

Dependências de dados

Falsas

- **Dependência de saída**

- **Exemplo**

- **add R1, R2, R3** : **$R3 = R1 + R2$;**
 - **sub R4, R3, R5** : **$R5 = R4 - R3$;**
 - **add R7, R8, R3** : **$R3 = R7 + R8$;**

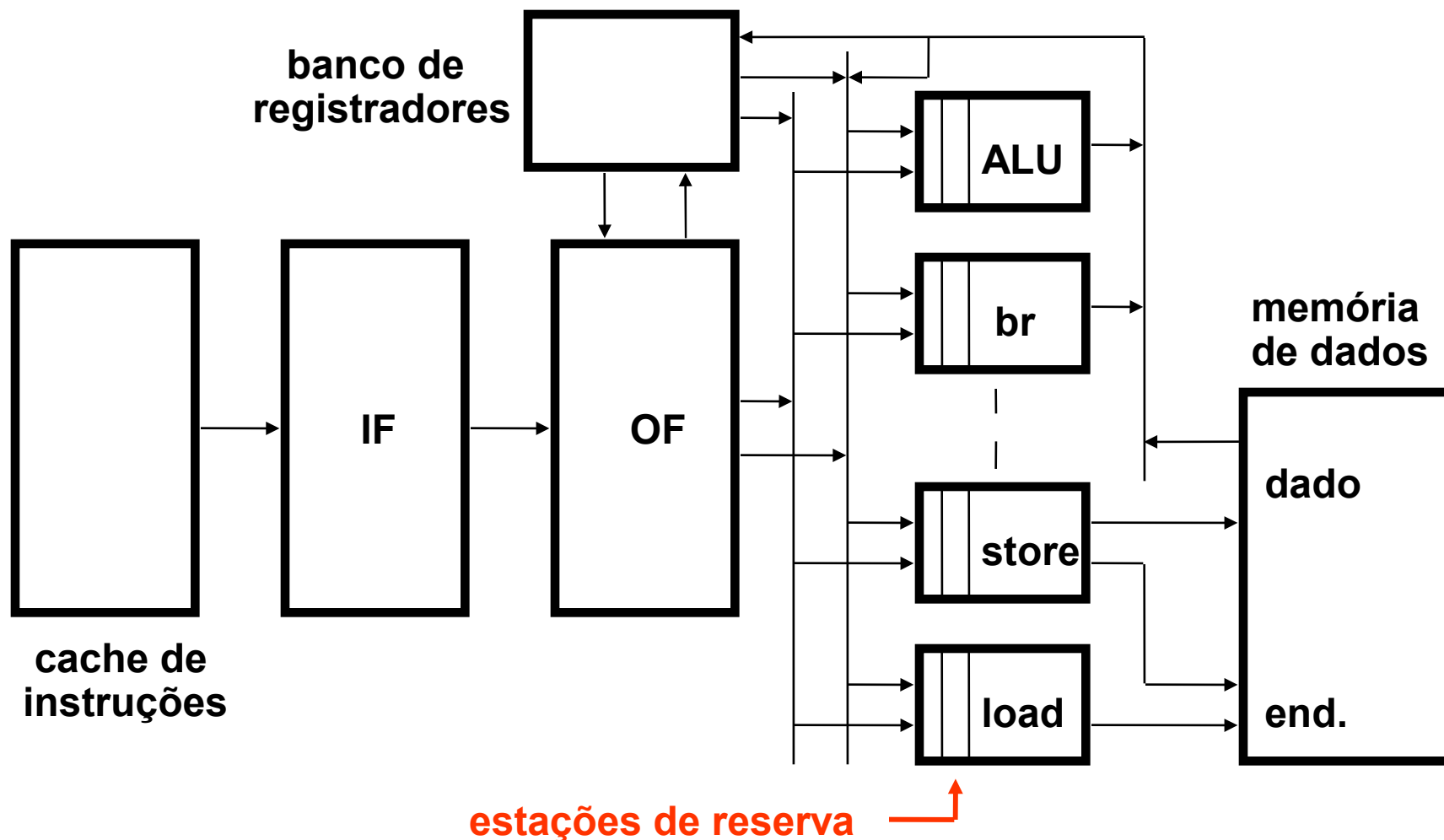
- **Instruções 1 e 3 escrevem no mesmo operando (R3)**
 - **Instrução 1 deve escrever em R3 antes da instrução 3, caso contrário será usado um valor errado de R3 na instrução 2**
 - **WAW (write-after-write)**

6. Janela de Instruções Distribuída

6. Janela de Instruções Distribuída

- Cada unidade de execução tem uma “**estação de reserva**”
 - estação com capacidade para armazenar 2 a 6 instruções
- Instruções são decodificadas e enviadas para a estação de reserva apropriada
- Instruções são enviadas para unidade de execução quando operandos estão disponíveis
- Mesmo mecanismo de identificação de registradores nas instruções
- Quando registradores são atualizados, valores são passados diretamente para as estações de reserva
 - busca associativa para substituição de identificadores por valores
- Algoritmo de Tomasulo: combinação de estações de reserva distribuídas com renomeação de registradores

Janela de Instruções Distribuída



7. Exemplo da execução de instruções em um Processador Superescalar com OoO

7. Exemplo da execução de instruções em um Processador Superescalar com OoO

- Supondo um processador superescalar com a seguinte configuração:
 - 4 unidades funcionais - 2 somadores, 1 multiplicador, 1 load/store
 - pode executar 4 instruções por ciclo em cada estágio do pipeline
 - latências
 - somador - 1 ciclo
 - multiplicador - 2 ciclos
 - load/store - 2 ciclos
- Deve ser executado o seguinte programa:

ADD R1, R2, R3

LW R10, 100 (R5)

ADD R5, R1, R6

MUL R7, R4, R8

ADD R2, R7, R3

ADD R9, R4, R10

ADD R11, R4, R6

Exemplo

ciclo	somador 1	somador 2	multiplicador	load/store
1	R1 = R2 + R3			
2				
3				

ADD R1, R2, R3

LW R10, 100 (R5)

ADD R5, R1, R6

MUL R7, R4, R8

ADD R2, R7, R3

ADD R9, R4, R10

ADD R11, R4, R6

 dependências verdadeiras

 dependências falsas

Exemplo

ciclo	somador 1	somador 2	multiplicador	load/store
1	R1 = R2 + R3			R10 = mem (R5+100)
2				R10 = mem (R5+100)
3				

ADD R1, R2, R3
LW R10, 100 (R5)
 ADD R5, R1, R6
 MUL R7, R4, R8
 ADD R2, R7, R3
 ADD R9, R4, R10
 ADD R11, R4, R6

 dependências verdadeiras

 dependências falsas

Exemplo

ciclo	somador 1	somador 2	multiplicador	load/store
1	R1 = R2 + R3			R10 = mem (R5+100)
2	R5 = R1 + R6			R10 = mem (R5+100)
3				

```

ADD  R1, R2, R3
LW   R10, 100 (R5)
ADD  R5, R1, R6
MUL  R7, R4, R8
ADD  R2, R7, R3
ADD  R9, R4, R10
ADD  R11, R4, R6
    
```

 dependências verdadeiras

 dependências falsas

Exemplo

ciclo	somador 1	somador 2	multiplicador	load/store
1	R1 = R2 + R3		R7 = R4 * R8	R10 = mem (R5+100)
2	R5 = R1 + R6		R7 = R4 * R8	R10 = mem (R5+100)
3				

```

ADD  R1, R2, R3
LW   R10, 100 (R5)
ADD  R5, R1, R6
MUL  R7, R4, R8
ADD  R2, R7, R3
ADD  R9, R4, R10
ADD  R11, R4, R6
    
```

 dependências verdadeiras

 dependências falsas

Exemplo

ciclo	somador 1	somador 2	multiplicador	load/store
1	$R1 = R2 + R3$		$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
2	$R5 = R1 + R6$		$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
3		$R2 = R7 + R3$		

ADD R1, R2, R3
 LW R10, 100(R5)
 ADD R5, R1, R6
 MUL R7, R4, R8
ADD R2, R7, R3
 ADD R9, R4, R10
 ADD R11, R4, R6

→ dependências verdadeiras

→ dependências falsas

Exemplo

ciclo	somador 1	somador 2	multiplicador	load/store
1	$R1 = R2 + R3$		$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
2	$R5 = R1 + R6$		$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
3	$R9 = R4 + R10$	$R2 = R7 + R3$		

ADD R1, R2, R3
 LW R10, 100(R5)
 ADD R5, R1, R6
 MUL R7, R4, R8
 ADD R2, R7, R3
ADD R9, R4, R10
 ADD R11, R4, R6

→ dependências verdadeiras

→ dependências falsas

Exemplo

ciclo	somador 1	somador 2	multiplicador	load/store
1	$R1 = R2 + R3$	$R11 = R4 + R6$	$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
2	$R5 = R1 + R6$		$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
3	$R9 = R4 + R10$	$R2 = R7 + R3$		

ADD R1, R2, R3
 LW R10, 100(R5)
 ADD R5, R1, R6
 MUL R7, R4, R8
 ADD R2, R7, R3
 ADD R9, R4, R10
ADD R11, R4, R6

→ dependências verdadeiras





→ dependências falsas

8. Renomeação de Registradores

Renomeação de Registradores

- Antidependências e dependências de saída são causadas pela reutilização de registradores
- Efeito destas dependências pode ser reduzido pelo aumento do número de registradores ou pela utilização de outros registradores disponíveis

- Exemplo

– ADD R1, R2, R3	;  R1 = R2 + R3	
– ADD R2, R1, 1	; R2 = R1 + 1	antidependência em R2 
– ADD R1, R4, R5	;  R1 = R4 + R5	dependência de saída em R1 

- Utilizando 2 outros registradores R6 e R7 pode-se eliminar as dependências falsas

– ADD R1, R2, R3	; R1 = R2 + R3
– ADD R6, R1, 1	; R6 = R1 + 1
– ADD R7, R4, R5	; R7 = R4 + R5

Renomeação de Registradores

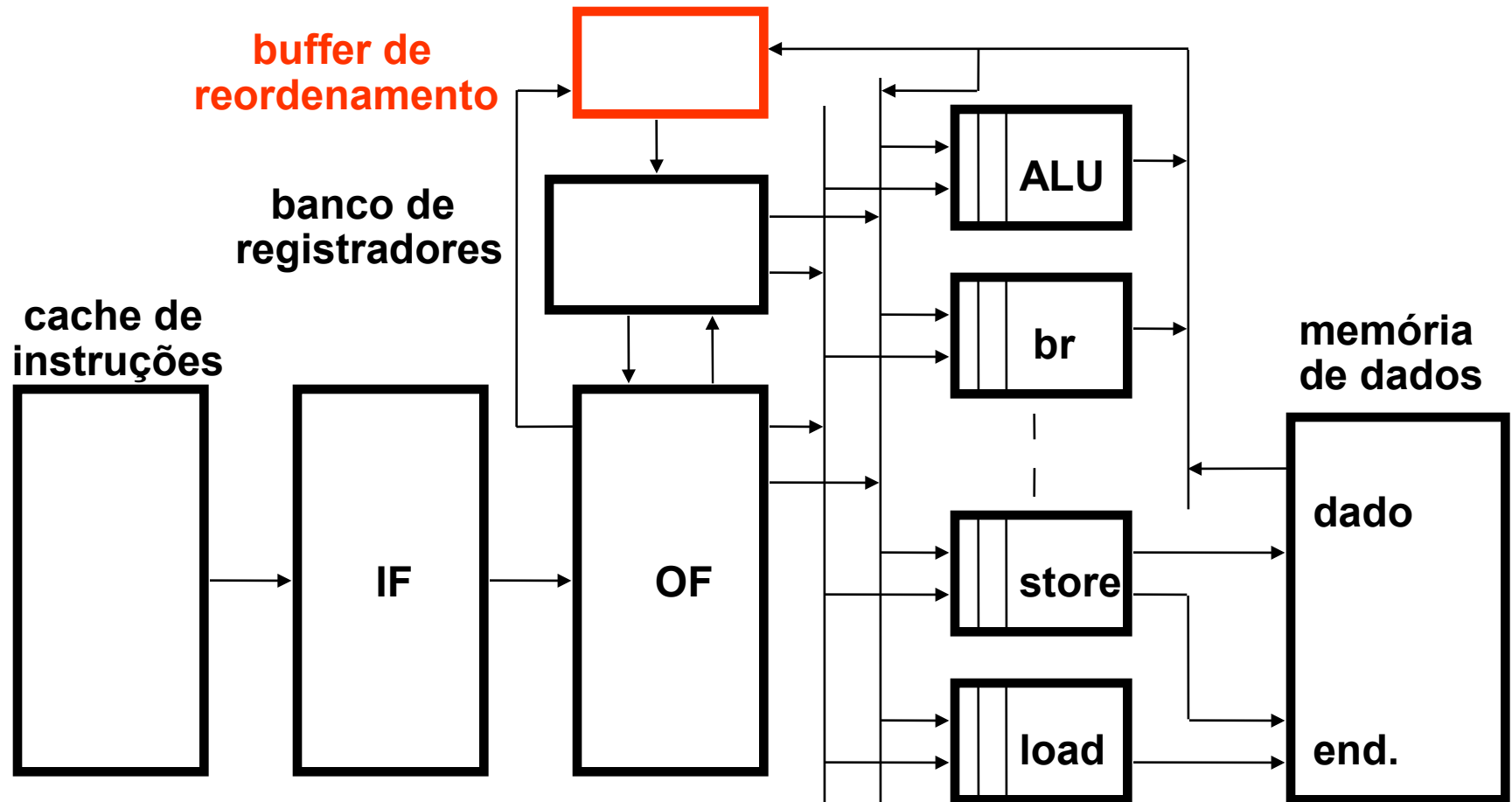
- **Não é possível criar número ilimitado de registradores**
- **Arquitetura deve manter compatibilidade quanto aos registradores visíveis para o programador**
- **Solução**
 - utilizar banco de registradores interno, bem maior do que o banco visível
 - renomear registradores temporariamente
 - cada registrador visível que é escrito numa instrução é renomeado para um registrador interno escolhido dinamicamente
- **No exemplo anterior, supondo registradores internos Ra, Rb, Rc, Rd, Re, Rf, Rg**
 - **ADD Ra, Rb, Rc**
 - **ADD Rd, Ra, 1**
 - **ADD Re, Rf, Rg**
- **Antidependência e dependência de saída foram eliminadas**

9. Reordenamento

Reordenamento

- A fila de reordenação é implementada numa FIFO e serve para estabelecer a reordenação das instruções executadas fora de ordem.
- Uma entrada é colocada nesta fila sempre que uma instrução é despachada e quando a instrução é executada o resultado é colocado na fila.
- Ao final de uma sequência de instruções, as entradas na fila são retiradas através da operação de gradação e os resultados armazenados nos registradores.

9. Buffer de Reordenamento



Buffer de Reordenamento

- **Buffer é organizado como FIFO**
- **Quando decodifica-se instrução que escreve em registrador, posição do buffer é alocada para o resultado**
- **Cada posição do buffer contém**
 - **número do registrador original**
 - **campo para armazenamento do resultado**
 - **tag de renomeação**
- **Quando o resultado está disponível, valor é escrito no buffer**
 - **valor é simultaneamente enviado para estações de reserva e substitui tag de renomeação correspondente, se encontrado**

FIM