

Arquitetura de Computadores III

Pipeline Superescalar de Instruções

Superescalaridade

- 1. Introdução
- 2. Despacho em ordem, terminação em ordem
- 3. Despacho em ordem, terminação fora-de-ordem
- 4. Despacho fora-de-ordem, terminação fora-de-ordem
- 5. Janela de instruções centralizada
- 6. Janela de instruções distribuída
- 7. Exemplo
- 8. Renomeação de registradores

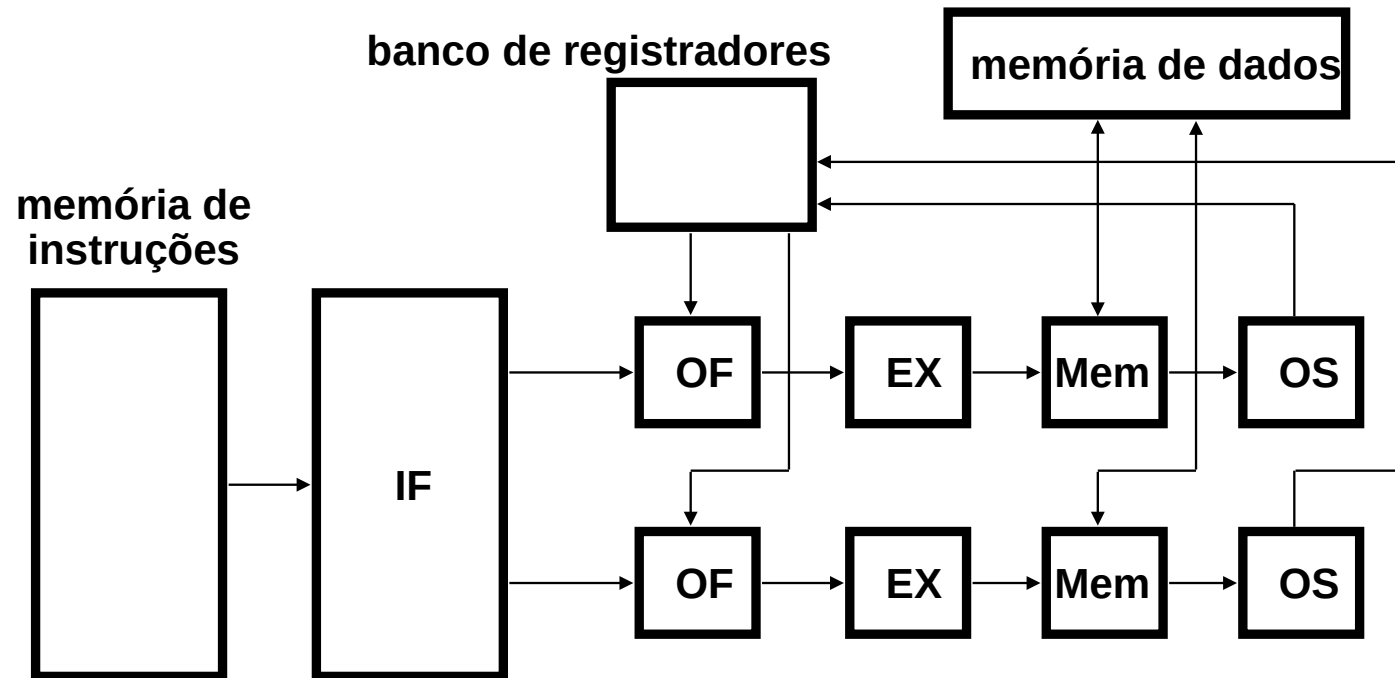
Introdução

- princípios da super-escalaridade
 - várias unidades de execução
 - várias instruções completadas simultaneamente em cada ciclo de relógio
- hardware é responsável pela extração de paralelismo
- na prática, obtém-se IPC pouco maior do que 2
 - limitação do paralelismo intrínseco dos programas
- problemas com a execução simultânea de instruções
 - conflitos de acesso a recursos comuns
 - memória
 - dependências de dados
 - verdadeiras
 - falsas - anti-dependências, dependências de saída
 - dependências de controle (desvios)

Introdução

- pipelines ou unidades funcionais podem operar com velocidades variáveis – latências
- término das instruções pode não seguir a seqüência estabelecida no programa
- processador com capacidade de “look-ahead”
 - se há conflito que impede execução da instrução atual, processador
 - examina instruções além do ponto atual do programa
 - procura instruções que sejam independentes
 - executa estas instruções
- possibilidade de execução fora de ordem
 - cuidado para manter a correção dos resultados do programa

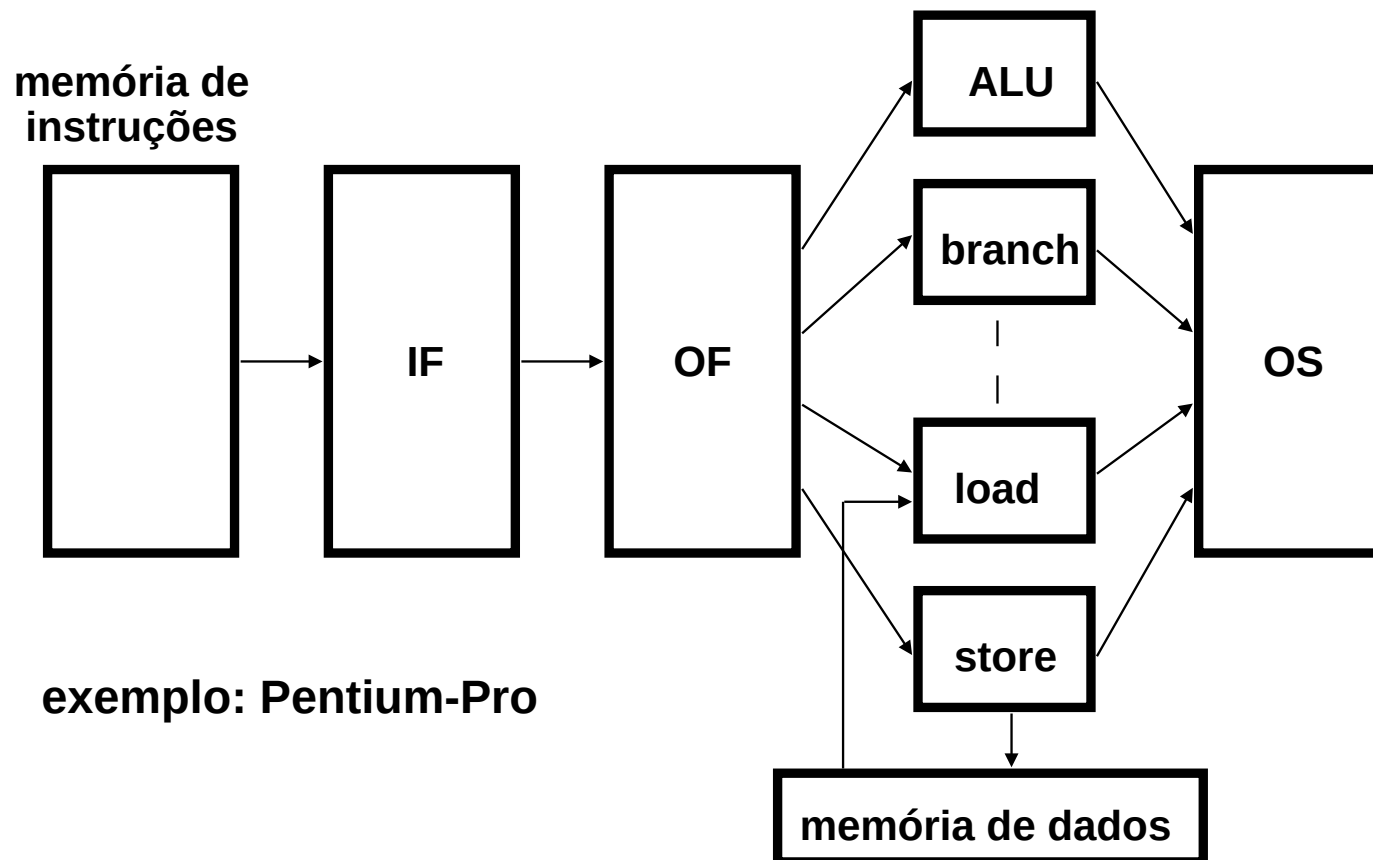
Processador com 2 pipelines



exemplo: Pentium I

cache de instruções precisa fornecer dobro de instruções por ciclo

Unidades de execução especializadas



Despacho e terminação de instruções

- despacho de instruções
 - refere-se ao fornecimento de instruções para as unidades funcionais
- terminação de instruções
 - refere-se à escrita de resultados (em registradores, no caso de processadores RISC)
- alternativas
 - despacho em ordem, terminação em ordem
 - despacho em ordem, terminação fora de ordem
 - despacho fora de ordem, terminação fora de ordem

Despacho em ordem, terminação em ordem

- despacho de novas instruções só é feito quando instruções anteriormente despachadas já foram executadas
- despacho é congelado ...
 - quando existe conflito por unidade funcional
 - quando unidade funcional exige mais de um ciclo para gerar resultado
- exemplo, supondo processador que pode a cada ciclo ...
 - decodificar 2 instruções
 - executar até 3 instruções em 3 unidades funcionais distintas
 - escrever resultados de 2 instruções

Despacho em ordem, terminação em ordem

decodificação	execução	write-back	ciclo
I1			1
I3	I1		2
I3			3
		I1	4
I5		I3	5
	I5		6
	I6	I5	7
		I6	8

restrições:

- fase de execução de I1 exige 2 ciclos
- I3 e I4 precisam da mesma unidade funcional
- I5 e I6 precisam da mesma unidade funcional
- I5 depende do valor produzido por I4

6 instruções em
6 ciclos
IPC = 1.0

Despacho em ordem, terminação fora de ordem

- despacho não espera que instruções anteriores já tenham sido executadas
 - ou seja: despacho não é congelado quando unidades funcionais levam mais de um ciclo para executar instrução
- consequência: uma unidade funcional pode completar uma instrução após instruções subseqüentes já terem sido completadas
- despacho ainda precisa ser congelado quando ...
 - há conflito por uma unidade funcional
 - há uma dependência de dados verdadeira

Despacho em ordem, terminação fora de ordem

decodificação

I1	I2
I3	I4
	I4
I5	I6
	I6

execução

I1	I2	
I1		I3
		I4
	I5	
	I6	

write-back

I2	
I1	I3
I4	
I5	
I6	

ciclo

1
2
3
4
5
6
7

**6 instruções em
5 ciclos
IPC = 1.2**

notar:

- I1 termina fora de ordem em relação a I2
- I3 é executada concorrentemente com último ciclo de execução de I1
- tempo total reduzido para 7 ciclos

Despacho em ordem, terminação fora de ordem

- supondo a seguinte situação
 - $R3 := R3 \text{ op } R5$
 - $R4 := R3 + 1$
 - $R3 := R5 + 1$
- dependência de saída
 - 1ª e 3ª instrução escrevem em R3
 - valor final de R3 deve ser o escrito pela 3ª instrução
 - atribuição da 1ª instrução não pode ser feita após atribuição da 3ª instrução
 - despacho da 3ª instrução precisa ser congelado
- terminação fora de ordem ...
 - exige controle mais complexo para testar dependências de dados
 - torna mais difícil o tratamento de interrupções

Despacho fora de ordem, terminação fora de ordem

- problemas do despacho em ordem
 - decodificação de instruções é congelada quando instrução cria ...
 - conflito de recurso
 - dependência verdadeira ou dependência de saída
 - consequência: processador não tem capacidade de look-ahead além da instrução que causou o problema, mesmo que haja instruções posteriores independentes
- solução
 - isolar estágio de decodificação do estágio de execução
 - continuar buscando e decodificando instruções, mesmo que elas não possam ser executadas imediatamente
 - inclusão de um buffer entre os estágios de decodificação e execução: janela de instruções
 - instruções são buscadas de janela independentemente de sua ordem de chegada: despacho fora de ordem

Despacho fora de ordem, terminação fora de ordem

decodificação	janela	execução	write-back	ciclo
I1 I2				1
I3 I4	I1, I2	I1 I2		2
I5 I6	I3, I4	I1 I3	I2	3
	I4, I5, I6	I6 I4	I1 I3	4
	I5	I5	I4 I6	5
			I5	6

6 instruções em
4 ciclos
IPC = 1.5

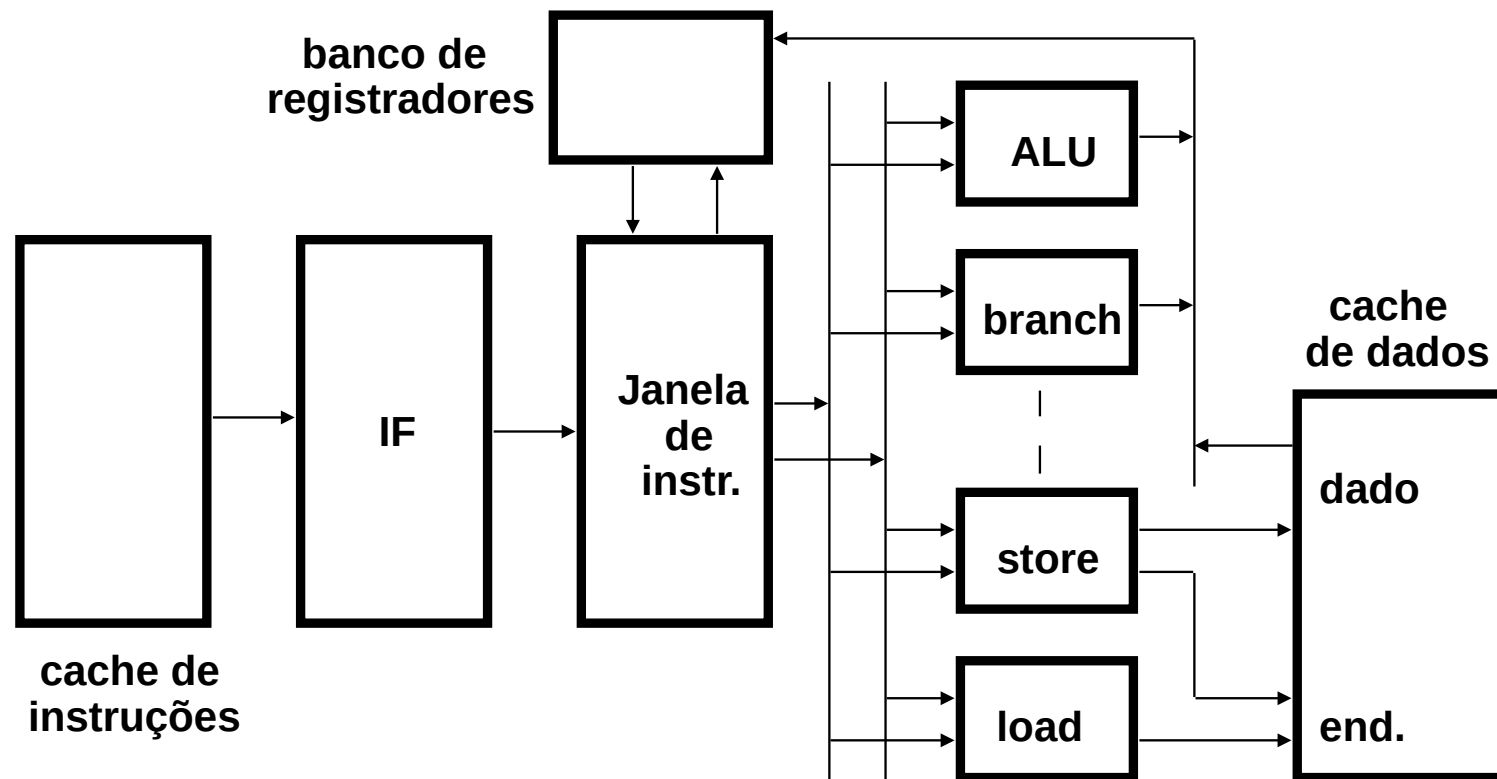
notar:

- estágio de decodificação opera a velocidade máxima, pois independe do estágio de execução
- I6 é independente e pode ser executada fora de ordem, concorrentemente com I4
- tempo total reduzido para 6 ciclos

Despacho fora de ordem, terminação fora de ordem

- supondo a seguinte situação
 - $R4 := R3 + 1$
 - $R3 := R5 + 1$
- anti-dependência
 - 2ª instrução escreve em R3
 - 1ª instrução precisa ler valor de R3 antes que 2ª instrução escreva novo valor
 - despacho da 2ª instrução precisa ser congelado até que 1ª instrução tenha lido valor de R3

Janela de instruções centralizada



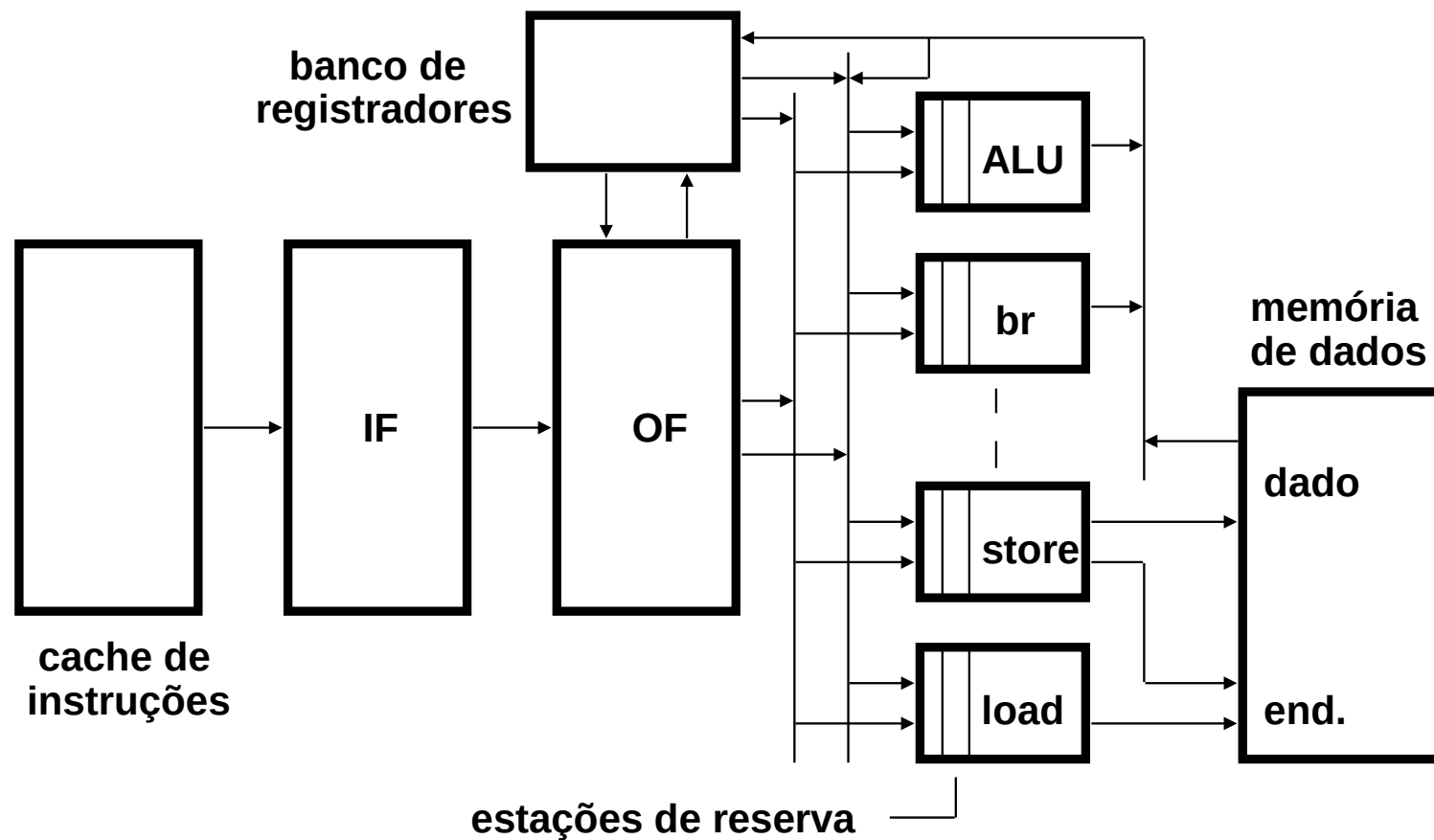
Janela de instruções centralizada

- “janela de instruções” é um buffer que armazena todas as instruções pendentes para execução
- instrução enviada para unidade de execução correspondente quando operandos estão disponíveis
 - operandos buscados no banco de registradores
- se operando não está disponível, identificador de registrador é colocado na instrução
 - quando instrução atualiza este registrador, janela de instruções é pesquisada associativamente e identificador do registrador é substituído pelo valor do operando

Janela de instruções centralizada

instr.	código	registr. destino	oper. 1	reg.1	oper. 2	reg.2
1	operação	ID	valor			ID
2	operação	ID	valor			ID
3	operação	ID		ID	valor	
4	operação	ID	valor		valor	
5	operação	ID		ID		ID

Janela de instruções distribuída



Janela de instruções distribuída

- cada unidade de execução tem uma “estação de reserva”
 - estação tem 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

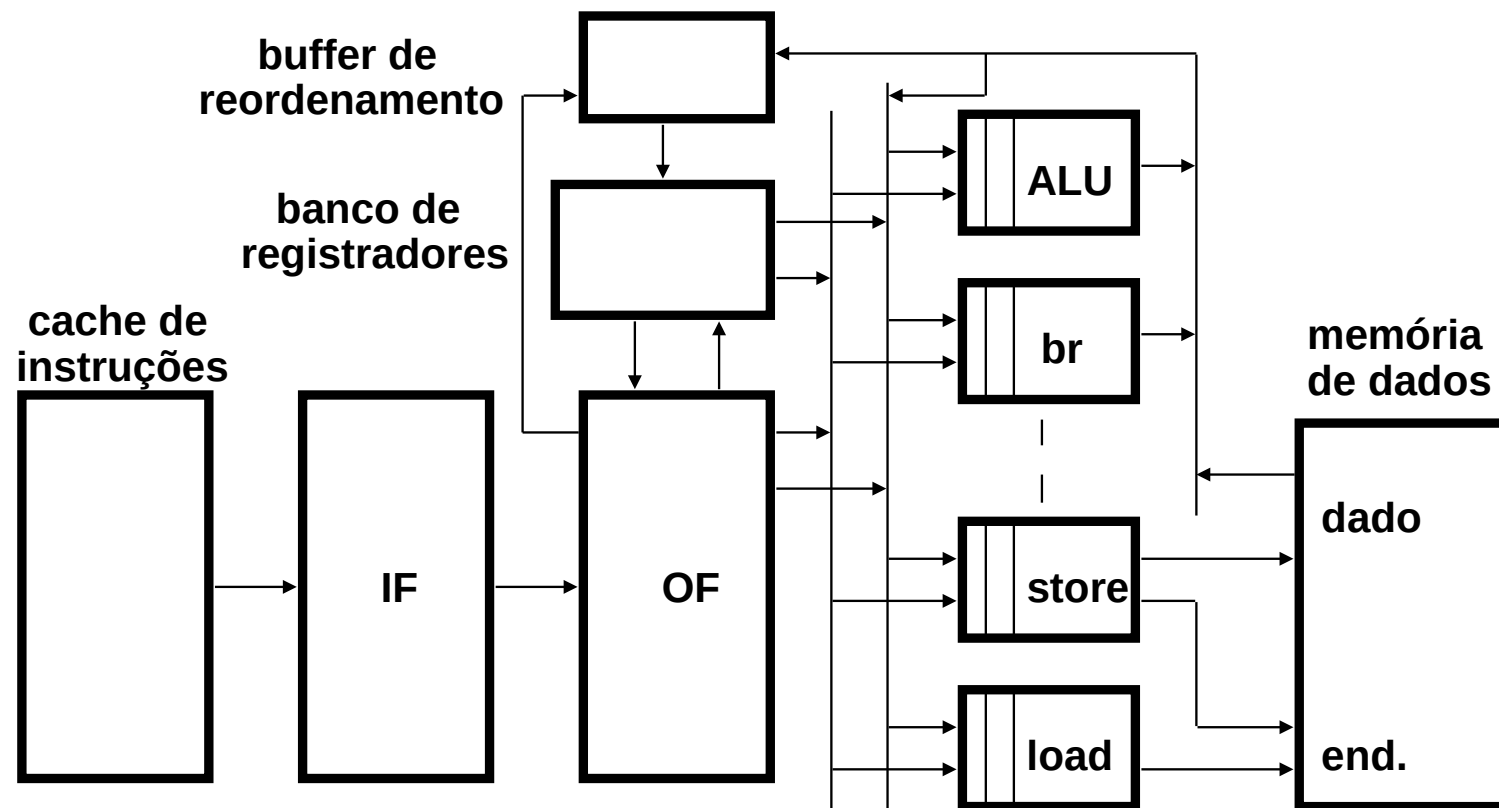
Renomeação de registradores

- 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, R8, 1 ; $R2 = R8 + 1$ 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, R8, 1 ; $R6 = R8 + 1$
 - 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, Rh
 - ADD Ra, Rb, Rc
 - ADD Rd, Rh, 1
 - ADD Re, Rf, Rg
- antidependência e dependência de saída foram eliminadas

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 resultado está disponível, valor é escrito no buffer
 - valor é simultaneamente enviado para estações de reserva e substitui tag de renomeação correspondente, se encontrado

Exemplo

- 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				

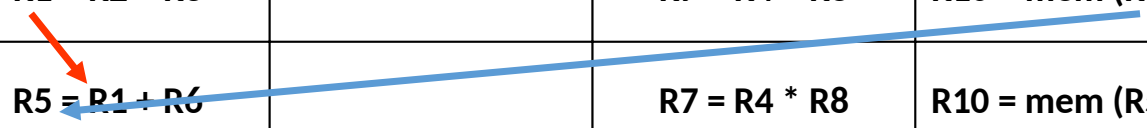
ADD R1, R2, R3
 LW R10, 100 (R5)
 ADD R5, R1, R6
 MUL R7, R4, R8
 ADD R2, R7, R3
 ADD R9, R4, R10
 ADD R11, R4, R6

→ 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 = mem (R5+100)
2	R5 = R1 + R6		R7 = R4 * R8	R10 = 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 = mem (R5+100)
2	R5 = R1 + R6		R7 = R4 * R8	R10 = 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

Exemplo

ciclo	somador 1	somador 2	multiplicador	load/store
1	R1 = R2 + R3	R11 = R4 + R6	R7 = R4 * R8	R10 = mem (R5+100)
2	RA = R1 + R6		R7 = R4 * R8	R10 = mem (R5+100)
3	R9 = R4 + R10	RB = R7 + R3		

```

ADD  R1, R2, R3
LW   R10, 100 (R5)
ADD  RA, R1, R6
MUL  R7, R4, R8
ADD  RB, R7, R3
SUB  R22, R23, R2
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	$RA = R1 + R6$		$R7 = R4 * R8$	$R10 = \text{mem}(R5+100)$
3	$R9 = R4 + R10$	$RB = R7 + R3$		

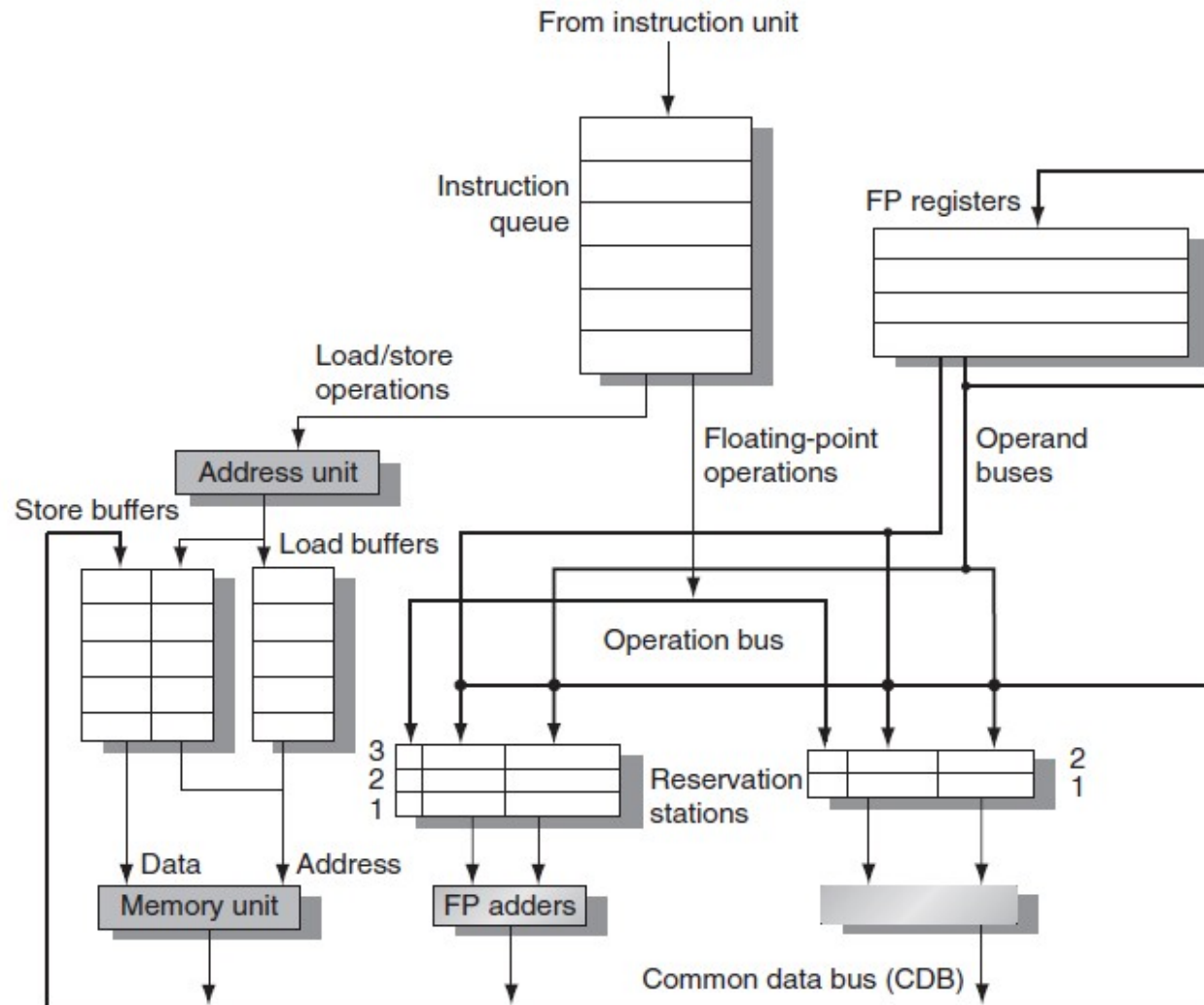
```

ADD  R1, R2, R3
LW   R10, 100(R5)
ADD  RA, R1, R6
MUL  R7, R4, R8
ADD  RB, R7, R3
SUB  R22, R23, RB
ADD  R9, R4, R10
ADD  R11, R4, R6
    
```

→ dependências verdadeiras

→ dependências falsas

Algoritmo de Tomasulo – Ponto flutuante no MIPS



HENNESSY, John L.
Arquitetura de computadores uma abordagem quantitativa

9/15/22

Arquitetura de Computadores III - Engenharia e Ciência da Computação - PUC Minas

102

Instruction		Instruction status		
		Issue	Execute	Write result
L.D	F6,32(R2)	✓	✓	✓
L.D	F2,44(R3)	✓	✓	
MUL.D	F0,F2,F4	✓		
SUB.D	F8,F2,F6	✓		
DIV.D	F10,F0,F6	✓		
ADD.D	F6,F8,F2	✓		

Reservation stations							
Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	No						
Load2	Yes	Load					44 + Regs[R3]
Add1	Yes	SUB		Mem[32 + Regs[R2]]	Load2		
Add2	Yes	ADD			Add1	Load2	
Add3	No						
Mult1	Yes	MUL		Regs[F4]	Load2		
Mult2	Yes	DIV		Mem[32 + Regs[R2]]	Mult1		

Register status									
Field	F0	F2	F4	F6	F8	F10	F12	...	F30
Qi	Mult1	Load2		Add2	Add1	Mult2			

Instruction		Instruction status		
		Issue	Execute	Write result
L.D	F6,32(R2)	✓	✓	✓
L.D	F2,44(R3)	✓	✓	
MUL.D	F0,F2,F4	✓		
SUB.D	F8,F2,F6	✓		
DIV.D		✓		
ADD.I		✓		

Reservation stations							
Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	No						
Load2	Yes	Load					44 + Regs[R3]
Add1	Yes	SUB		Mem[32 + Regs[R2]]	Load2		
Add2	Yes	ADD			Add1	Load2	
Add3	No						
Mult1	Yes	MUL		Regs[F4]	Load2		
Mult2	Yes	DIV		Mem[32 + Regs[R2]]	Mult1		

Register status									
Field	F0	F2	F4	F6	F8	F10	F12	...	F30
Qi	Mult1	Load2		Add2	Add1	Mult2			

Estação de reserva está ocupada?

Instruction		Instruction status		
		Issue	Execute	Write result
L.D	F6,32(R2)	✓	✓	✓
L.D	F2,44(R3)	✓	✓	
MUL.D	F0,F2,F4	✓		
SUB.D	F8,F2,F6	✓		
DIV.D		✓		
ADD.I		✓		

Operação que será realizada!

Reservation stations							
Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	No						
Load2	Yes	Load					44 + Regs[R3]
Add1	Yes	SUB		Mem[32 + Regs[R2]]	Load2		
Add2	Yes	ADD			Add1	Load2	
Add3	No						
Mult1	Yes	MUL		Regs[F4]	Load2		
Mult2	Yes	DIV		Mem[32 + Regs[R2]]	Mult1		

Register status									
Field	F0	F2	F4	F6	F8	F10	F12	...	F30
Qi	Mult1	Load2		Add2	Add1	Mult2			

Instruction		Instruction status		
		Issue	Execute	Write result
L.D	F6,32(R2)	✓	✓	✓
L.D	F2,44(R3)	✓	✓	
MUL.D	F0,F2,F4	✓		
SUB.D	F8,F2,F6	✓		
DIV.D	F10,F0,F6	✓		
ADD.D	F6,F8,F2	✓		

Estações de reserva que produzirão o resultado!

Reservation stations							
Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	No						
Load2	Yes	Load					44 + Regs[R3]
Add1	Yes	SUB		Mem[32 + Regs[R2]]	Load2		
Add2	Yes	ADD			Add1	Load2	
Add3	No						
Mult1	Yes	MUL		Regs[F4]	Load2		
Mult2	Yes	DIV		Mem[32 + Regs[R2]]	Mult1		

Register status									
Field	F0	F2	F4	F6	F8	F10	F12	...	F30
Qi	Mult1	Load2		Add2	Add1	Mult2			

Instruction		Instruction status		
		Issue	Execute	Write result
L.D	F6,32(R2)	✓	✓	✓
L.D	F2,44(R3)	✓	✓	
MUL.D	F0,F2,F4	✓		
SUB.D	F8,F2,F6	✓		
DIV.D	F10,F0,F6			
ADD.D	F6,F8,F2			

Valor dos operandos fonte!

Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	No						
Load2	Yes	Load					44 + Regs[R3]
Add1	Yes	SUB		Mem[32 + Regs[R2]]	Load2		
Add2	Yes	ADD			Add1	Load2	
Add3	No						
Mult1	Yes	MUL		Regs[F4]	Load2		
Mult2	Yes	DIV		Mem[32 + Regs[R2]]	Mult1		

Register status									
Field	F0	F2	F4	F6	F8	F10	F12	...	F30
Qi	Mult1	Load2		Add2	Add1	Mult2			

Instruction		Instruction status		
		Issue	Execute	Write result
L.D	F6,32(R2)	✓	✓	✓
L.D	F2,44(R3)	✓	✓	
MUL.D	F0,F2,F4	✓		
SUB.D	F8,F2,F6	✓		
DIV.D	F10,F0,F6	✓		
ADD.D	F6,F8,F2	✓		

1º: Valor imediato
2º: Endereço calculado

Reservation stations							
Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	No						
Load2	Yes	Load					44 + Regs[R3]
Add1	Yes	SUB		Mem[32 + Regs[R2]]	Load2		
Add2	Yes	ADD			Add1	Load2	
Add3	No						
Mult1	Yes	MUL		Regs[F4]	Load2		
Mult2	Yes	DIV		Mem[32 + Regs[R2]]	Mult1		

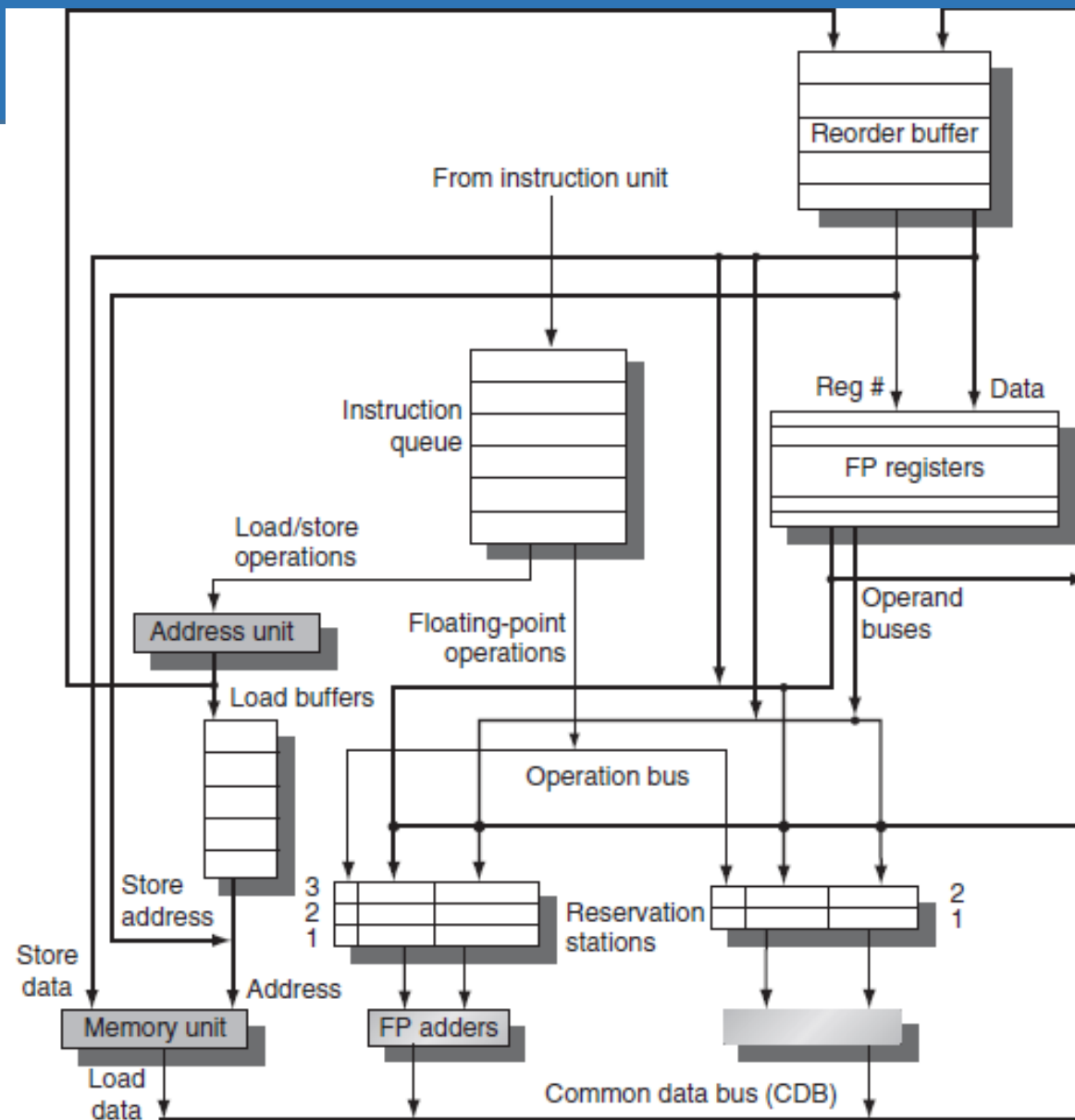
Register status									
Field	F0	F2	F4	F6	F8	F10	F12	...	F30
Qi	Mult1	Load2		Add2	Add1	Mult2			

Instruction		Instruction status		
		Issue	Execute	Write result
L.D	F6,32(R2)	✓	✓	✓
L.D	F2,44(R3)	✓	✓	
MUL.D	F0,F2,F4	✓		
SUB.D	F8,F2,F6	✓		
DIV.D	F10,F0,F6	✓		
ADD.D	F6,F8,F2	✓		

Reservation stations							
Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	No						
Load2	Yes	Load					44 + Regs[R3]
Add1	Yes	SUB		Mem[32 + Regs[R2]]	Load2		
Add2	Yes	ADD			Add1	Load2	
				Regs[F4]	Load2		
				Mem[32 + Regs[R2]]	Mult1		

Nome da estação de reserva
que fornecerá o resultado ao
registrador

Register status									
Field	F0	F2	F4	F6	F8	F10	F12	...	F30
Qi	Mult1	Load2		Add2	Add1	Mult2			



HENNESSY, John L.
Arquitetura de computadores uma abordagem quantitativa

9/15/22

Reorder buffer						
Entry	Busy	Instruction		State	Destination	Value
1	No	L.D	F6, 32 (R2)	Commit	F6	Mem[32 + Regs[R2]]
2	No	L.D	F2, 44 (R3)	Commit	F2	Mem[44 + Regs[R3]]
3	Yes	MUL.D	F0, F2, F4	Write result	F0	#2 × Regs[F4]
4	Yes	SUB.D	F8, F2, F6	Write result	F8	#2 − #1
5	Yes	DIV.D	F10, F0, F6	Execute	F10	
6	Yes	ADD.D	F6, F8, F2	Write result	F6	#4 + #2

Reservation stations								
Name	Busy	Op	Vj	Vk	Qj	Qk	Dest	A
Load1	No							
Load2	No							
Add1	No							
Add2	No							
Add3	No							
Mult1	No	MUL.D	Mem[44 + Regs[R3]]	Regs[F4]			#3	
Mult2	Yes	DIV.D		Mem[32 + Regs[R2]]	#3		#5	

FP register status										
Field	F0	F1	F2	F3	F4	F5	F6	F7	F8	F10
Reorder #	3						6		4	5
Busy	Yes	No	No	No	No	No	Yes	...	Yes	Yes