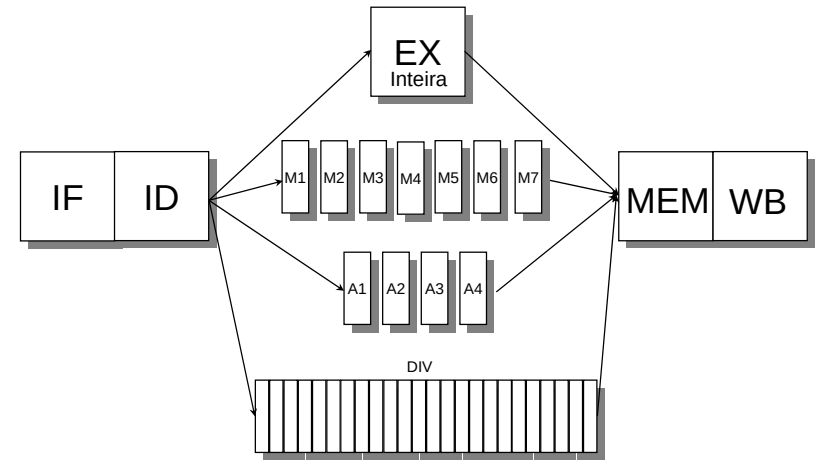


Aula 09: Escalonamento Dinâmico de Instruções, Scoreboarding

MIPS com Unidades de Execução c/ Pipeline Interno



Resumo

- Pipeline do MIPS:
 - busca de instrução em IF
 - *issue* em ID
 - checa antes hazards estruturais, RAWs e WAWs
 - *forward* para reduzir efeitos de *stall*
- No compilador:
 - escalonamento de código para evitar stalls no pipeline
 - *loop unrolling* para aumentar ILP em um bloco básico
- O que podemos querer agora?
 - Instrução entrando no pipeline ainda pode parar toda a execução ➡ e se dispararmos a instrução para execução fora de ordem?

Vantagens do Disparo de Instruções Fora de Ordem

- Como DLX possui múltiplas unidades funcionais para ponto flutuante, podemos ter mais chance de manter pipeline cheio
- Funciona até mesmo se compilador não puder escalonar o código (ex: fora de um bloco básico)
- Performance de código compilado fica mais independente de compilador
- Compilador é mais simples

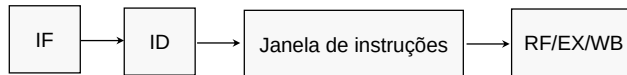
Disparo de Instruções Fora de Ordem

DIVD F0, F2, F4

ADDD F10, F0, F8

SUBD
F12, F8, F14

- DIVD executa em 24 ciclos
- ADDD está presa em ID aguardando DIVD completar
- SUBD está em IF aguardando a instrução que está em ID ser disparada



Disparo de Instruções Fora de Ordem

- Quebra de ID em duas fases:
 - Verificação de hazards estruturais: *in-order issue*
 - Espera por resultados (hazards de dados): execução fora de ordem, i.e., na medida em que os dados vão chegando, as instruções começam a executar
- Problemas com *Out-of-Order Issue*:
 - O que acontece agora com interrupções?

Funcionamento do Pipeline com Disparo Fora de Ordem

- Issue (primeira fase de ID)
- Lê Operandos (segunda fase de ID)
- Execução

Funcionamento do Pipeline com Disparo Fora de Ordem

- Issue (primeira fase de ID) :
 - Decodificação da instrução, verificação de hazards estruturais
 - Instrução vai ou para buffer de instruções pendentes ou para próximo estágio (no DLX, $|\text{buffer}| = 1$, i.e., somente uma instrução podia ficar pendente em ID)
- Lê Operandos (segunda fase de ID):
 - Instrução que foi permitida executar entra em modo de espera hazard de dados sejam liberados
 - Se buffer não estiver vazio, instrução vem de buffer, caso contrário, vem de registrador entre Issue e Lê Operandos

Funcionamento do Pipeline com Disparo Fora de Ordem

- Lê Operandos (segunda fase de ID):
 - Instrução que foi permitida executar entra em modo de espera hazard de dados sejam liberados
 - Se buffer não estiver vazio, instrução vem de buffer, caso contrário, vem de registrador entre Issue e Lê Operandos

Funcionamento do Pipeline com Disparo Fora de Ordem

- Execução:
 - Mantemos três estados para cada instrução sendo executada: Iniciando, Executando e Concluindo

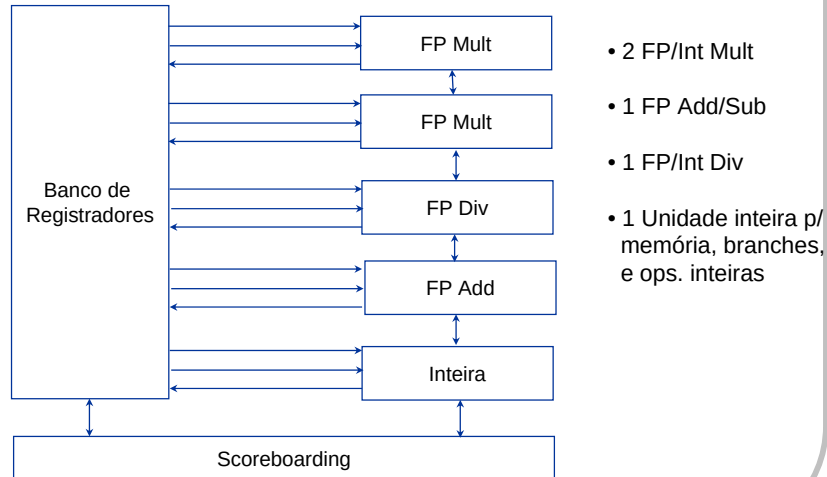
Scoreboarding

- Uma implementação de escalonamento dinâmica
 - Usada inicialmente no CDC-6600
- **Permite uma instrução executar fora de ordem quando existirem recursos suficientes e nenhuma dependência de dados**
 - WAR poderá ocorrer porque uma instrução posterior poderá ser disparada antes que uma instrução anterior possa ler operandos
- Objetivo final: CPI = 1 instr/ciclo

Scoreboarding Estágios do Pipeline do MIPS

- O pipeline possuirá estágios extras, até mesmo ignorando LD/ST, e ciclos adicionais para liberação de dependências no scoreboard
 - **IF**: busca de instrução
 - **ID**: decodificação, para o pipeline se houver hazard estrutural ou WAW
 - **SB**: scoreboard. Aguarda liberação de hazards do tipo RAW
 - **RF**: lê operandos do banco de registradores
 - **EX**: executa instrução em unidade funcional
 - **WB**: aguarda até que hazards WAR não possam ocorrer, e escreve resultado no banco de registradores

Diagrama do Processador



Scoreboarding: Exemplo

MULTDF0, F2, F4 ; 4 CICLOS
SD F0, 0(R4) ; 1 CICLO
ADDU R4, R2, #8 ; 1 CICLO
NEG F0, F4 ; 1 CICLO

	1	2	3	4	5	6	7	8	9	10	11	12	13
MULTD	IF	ID	SB	RF	EX	EX	EX	EX	WB				
SD		IF	ID	SB	SB	SB	SB	SB	RF	EX	WB		
ADDU			IF	ID	SB	RF	EX	-	-	WB			
NEG				IF	ID	ID	ID	ID	ID	SB	RF	EX	WB

Scoreboarding

Instruction Status

Instruction	Issued?	Operands read?	Exec. Complete?	Result written?

FU Status

Functional unit	Busy?	Op	Dest Reg	Source 1 reg	Source 1 from FU	Source 1 ready?	Source 2 reg	Source 2 from FU	Source 2 ready?

Pending Registers

	F0	F2	F4	F6	F8	F10	F12	F14	F16
FU generating result									

Regras para o Scoreboarding

- **Issue**
 - **SE:** Se FU não estiver ocupada (hazards estruturais) e não houver nenhum resultado pendente no registrador destino (hazards do tipo WAW)
 - **ENTÃO:** Aloque FU para executar instrução
 - “From FU” é assinalado entrada de “FU generating result”
 - “ready?” (significando pronto, mas ainda não lido) = TRUE somente se não existir nenhuma outra FU gerando o operando
- **Lê Operandos**
 - **SE:** Source 1 e Source 2 estão prontos (“ready?” = TRUE), nenhum hazard RAW
 - **ENTÃO:** “ready?” = FALSO para ambos, lê operandos, começa execução da instrução

Regras para o Scoreboarding

- **Writeback**
 - *SE: Nenhuma outra unidade funcional precisa ler o registrador usado como destino da operação e ainda não o leu*
 - *ENTÃO: Faça “ready” os operandos de qualquer outra FU que dependa desse resultado, limpe o campo “FU generating result”, e escreva o resultado no registrador*

Scoreboarding: Exemplo

LD F6, 34(R2)
LD F2, 45(R3)
MULTDF0, F2, F4
SUBD F8, F6, F2
DIVD F10, F0, F6
ADDD F6, F8, F2

- FP Add - 2 ciclos
- FP Mult - 10 ciclos
- FP Div - 20 ciclos

Situação inicial: Já completamos o primeiro LD, e estamos prontos para escrever o resultado do segundo LD

Scoreboarding

Instruction Status

Instruction	Issued?	Operands read?	Exec. Complete?	Result written?
LD F6, 34(R2)	x	x	x	x
LD F2, 45(R3)	x	x	x	
MULTD F0, F2, F4	x			
SUBD F8, F6, F2	x			
DIVD F10, F0, F6	x			
ADDD F6, F8, F2				

FU Status

Functional unit	Busy?	Op	Dest Reg	Source 1 reg	Source 1 from FU	Source 1 ready?	Source 2 reg	Source 2 from FU	Source 2 ready?
Integer	yes	LD	F2	R3	-	-	-	-	-
Mult1	yes	MULTD	F0	F2	Integer	no	F4	-	yes
Mult2	no	-	-	-	-	-	-	-	-
Add	yes	SUBD	F8	F6	-	yes	F2	Integer	no
Div	yes	DIVD	F10	F0	Mult1	no	F6	-	yes

Pending Registers

	F0	F2	F4	F6	F8	F10	F12	F14	F16
FU generating result	Mult1	Integer			Add	Div			

Scoreboarding: Exemplo

LD F6, 34(R2)
LD F2, 45(R3)
MULTD F0, F2, F4
SUBD F8, F6, F2
DIVD F10, F0, F6
ADDD F6, F8, F2

- FP Add - 2 ciclos
- FP Mult - 10 ciclos
- FP Div - 20 ciclos

MULTD acabou a execução e está pronta para escrever o resultado em F0

Scoreboarding

Instruction Status

Instruction	Issued?	Operands read?	Exec. Complete?	Result written?
LD F6, 34(R2)	x	x	x	x
LD F2, 45(R3)	x	x	x	x
MULTD F0, F2, F4	x	x	x	
SUBD F8, F6, F2	x	x	x	x
DIVD F10, F0, F6	x			
ADD F6, F8, F2	x	x	x	

FU Status

Functional unit	Busy?	Op	Dest Reg	reg	Source 1 from FU	ready?	reg	Source 2 from FU	ready?
Integer	no	-	-	-	-	-	-	-	-
Mult1	yes	MULTD	F0	F2	Integer	no	F4	-	no
Mult2	no	-	-	-	-	-	-	-	-
Add	yes	ADD	F6	F8	-	no	F2	-	no
Div	yes	DIVD	F10	F0	Mult1	no	F6	-	yes

Pending Registers

	F0	F2	F4	F6	F8	F10	F12	F14	F16
FU generating result	Mult1			Add		Div			

Scoreboarding: Exemplo

LD F6, 34(R2)

LD F2, 45(R3)

MULTD F0, F2, F4

SUBD F8, F6, F2

DIVD F10, F0, F6

ADD F6, F8, F2

- FP Add - 2 ciclos
- FP Mult - 10 ciclos
- FP Div - 20 ciclos

DIVD acabou a execução e está pronta para escrever o resultado em F10

Scoreboarding

Instruction Status

Instruction	Issued?	Operands read?	Exec. Complete?	Result written?
LD F6, 34(R2)	x	x	x	x
LD F2, 45(R3)	x	x	x	x
MULTD F0, F2, F4	x	x	x	x
SUBD F8, F6, F2	x	x	x	x
DIVD F10, F0, F6	x	x	x	
ADD F6, F8, F2	x	x	x	x

FU Status

Functional unit	Busy?	Op	Dest Reg	reg	Source 1 from FU	ready?	reg	Source 2 from FU	ready?
Integer	no	-	-	-	-	-	-	-	-
Mult1	no	-	-	-	-	-	-	-	-
Mult2	no	-	-	-	-	-	-	-	-
Add	no	-	-	-	-	-	-	-	-
Div	yes	DIVD	F10	F0	Mult1	no	F6	-	no

Pending Registers

	F0	F2	F4	F6	F8	F10	F12	F14	F16
FU generating result						Div			

Resumo de Scoreboarding

- Permite execução de instrução fora de ordem para operações não dependentes
- Tomada de decisão é centralizada:
 - Scoreboarding é do tamanho de uma unidade funcional
- Requer muitos barramentos para/de banco de registradores (hazard estrutural?)
- Não aproveita forwarding, mas impacto é de somente 1 ciclo, já que a escrita é feita tão logo quanto resultado fique pronto

Resumo de Scoreboarding

- Paralelismo é limitado por:
 - Janela de instruções (dependente do número de unidades funcionais e buffer)
 - Dependência de dados do programa (inerente da aplicação)
- Próxima aula
 - Register renaming
 - Algoritmo de Tomasulo