

# DCC007 – Organização de Computadores II

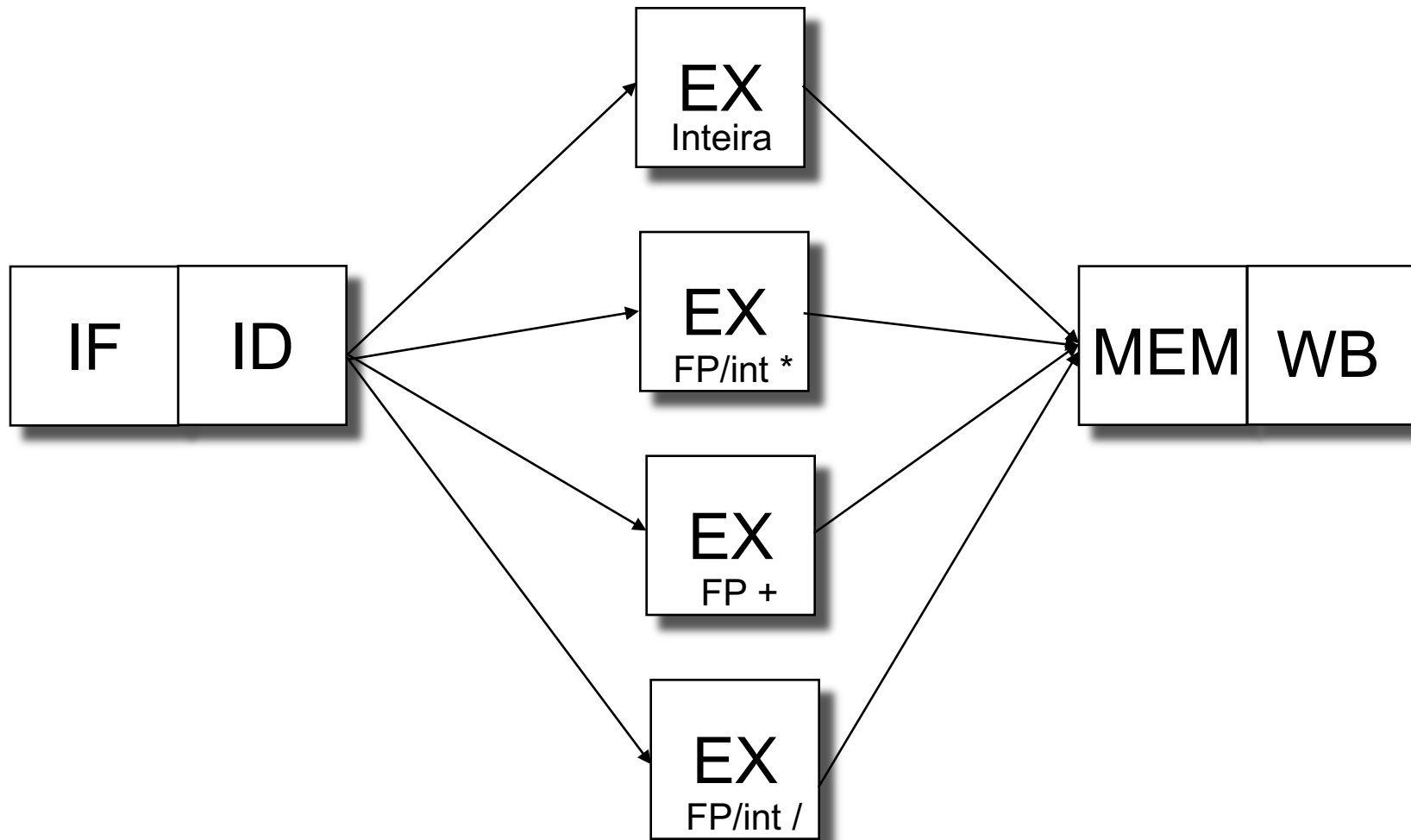
## **Aula 6 – Execução Multiciclo e MIPS R4000**

**Prof. Omar Paranaíba Vilela Neto**

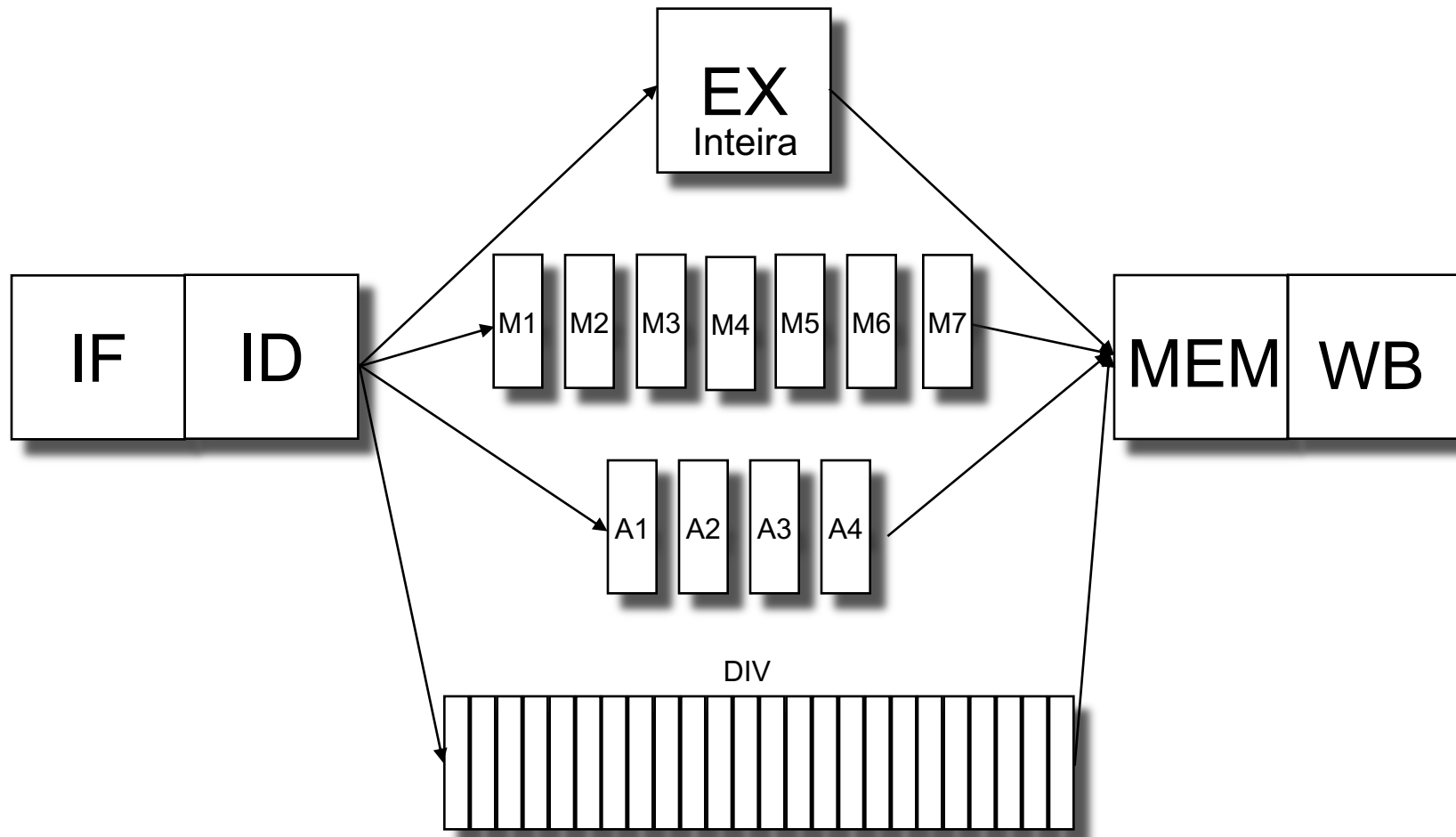


# Pipeline do MIPS com Instruções de Múltiplos Ciclos

---



# MIPS com Unidades de Execução c/ Pipeline Interno



# Diagrama do Pipeline

MULT.D	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM
ADD.D		IF	ID	A1	A2	A3	A4	MEM	WB	
L.D			IF	ID	EX	MEM	WB			
S.D				IF	ID	EX	MEM	WB		

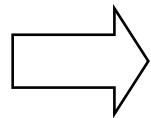
## Problemas com o Pipeline:

- Hazards estruturais
- Instruções podem completar em ordem diferente da ordem de emissão
- Número de escritas de registradores pode ser maior que 1 por ciclo
- WAW e WAR podem ocorrer
- Stalls serão mais frequentes
- E o que acontece com *forwardings*?

# Simplificação do Pipeline do MIPS

---

- Dois bancos de registradores ( $R_i$  e  $F_i$ )
- Instruções inteiras só podem acessar  $R_i$
- Instruções de ponto flutuante só podem acessar  $F_i$
- Instruções de load/store e de movimentação de dados fazem a comunicação entre  $R_i$  e  $F_i$



Simplifica lógica de detecção de hazard e forwarding

# Deteção de Hazards em ID

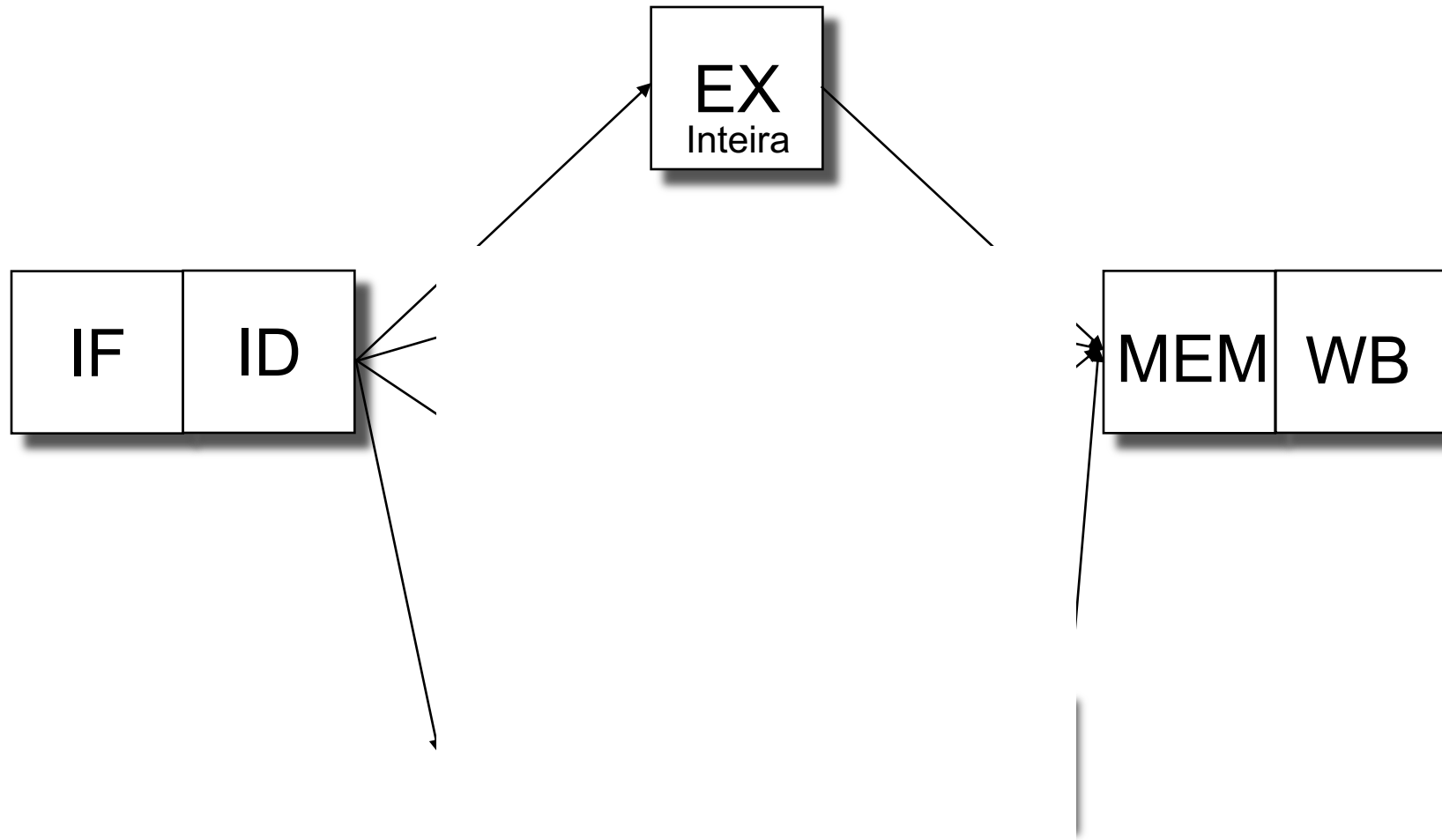
---

- Verificação de hazards estruturais: aguarde até unidades funcionais estarem disponíveis
- Verificação de hazards do tipo RAW: aguarde até que os operandos não estejam mais listados como pendências nas respectivas unidades funcionais
- Verificação de hazards do tipo WAW: determine se alguma instrução em  $A1, \dots, A4, D, M1, \dots, M7$  possui o mesmo registrador que algum operando em ID

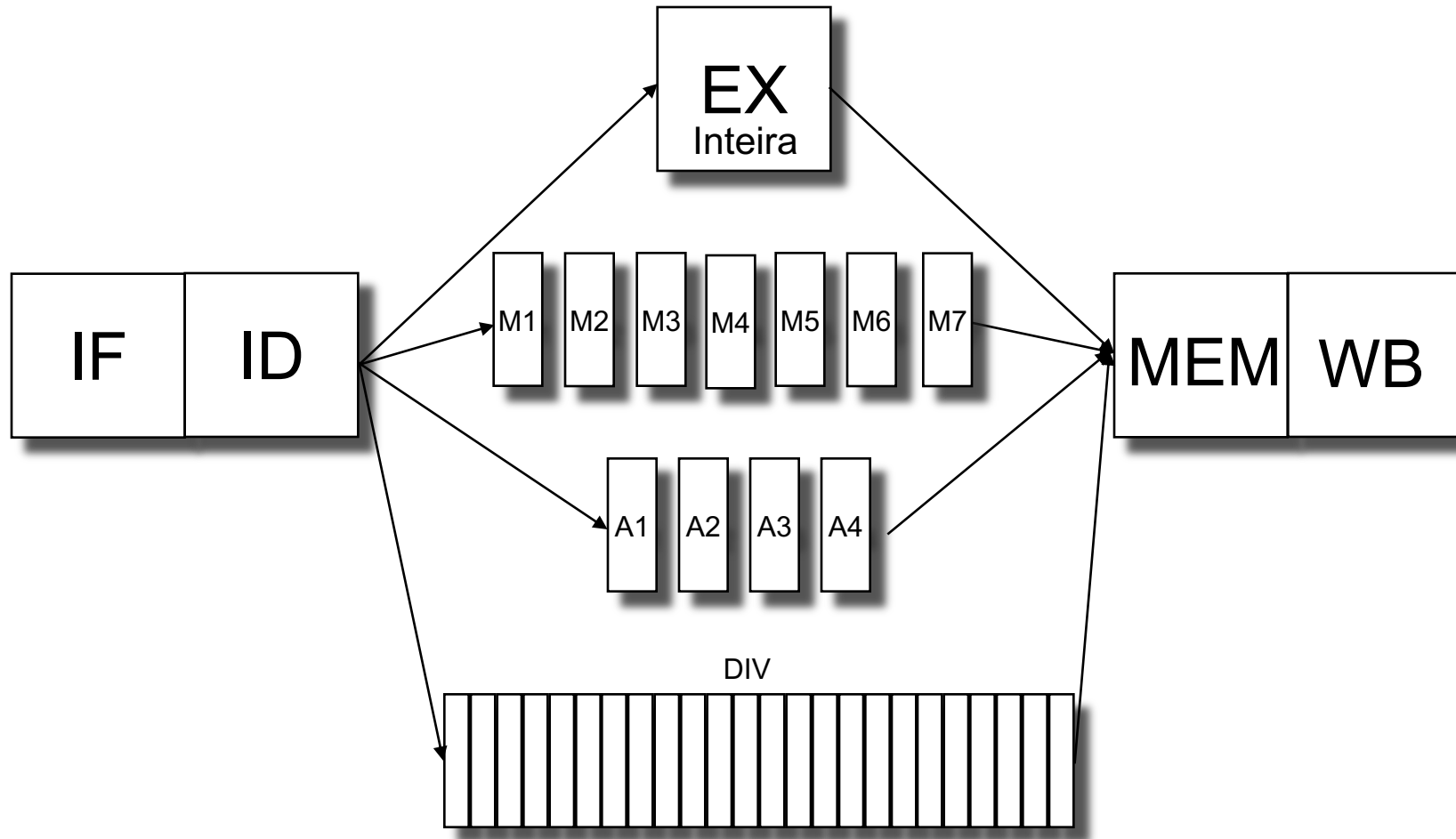
Só assim podemos iniciar a execução da instr!!!

# Lógica de Forwarding

---



# Lógica de Forwarding





# Lógica de Forwarding

---

- Só precisamos **checar** a mais se **registrador usado como resultado** é algum entre EX/MEM, A4/MEM, M7/MEM, D/MEM ou MEM/WB e **operando é um registrador de ponto flutuante** Fi

# Como Esse Código é Executado?

```
L.D      F4, 0(R2)
MUL.D    F0, F4, F6
ADD.D    F2, F0, F8
S.D      F2, 0(R2)
```

Instruction	Clock cycle number																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
L.D F4,0(R2)	IF	ID	EX	MEM	WB												
MUL.D F0,F4,F6		IF	ID	stall	M1	M2	M3	M4	M5	M6	M7	MEM	WB				
ADD.D F2,F0,F8			IF	stall	ID	stall	stall	stall	stall	stall	stall	A1	A2	A3	A4	MEM	WB
S.D F2,0(R2)					IF	stall	stall	stall	stall	stall	stall	ID	EX	stall	stall	stall	MEM

# Interrupções no Pipeline

---

- Observe sequência de código

DIV.D F0 , F2 , F4

ADD.D F10 , F10 , F8

SUB.D F12 , F12 , F14 ← Perda de precisão

- ADD.D já vai ter completado, mais DIV.D não (*out-of-order completion*)
  - O que fazer?
    - Operações de ponto flutuante podem destruir operandos
    - Término fora de ordem

# Solução para Manter Interrupções Precisas

---

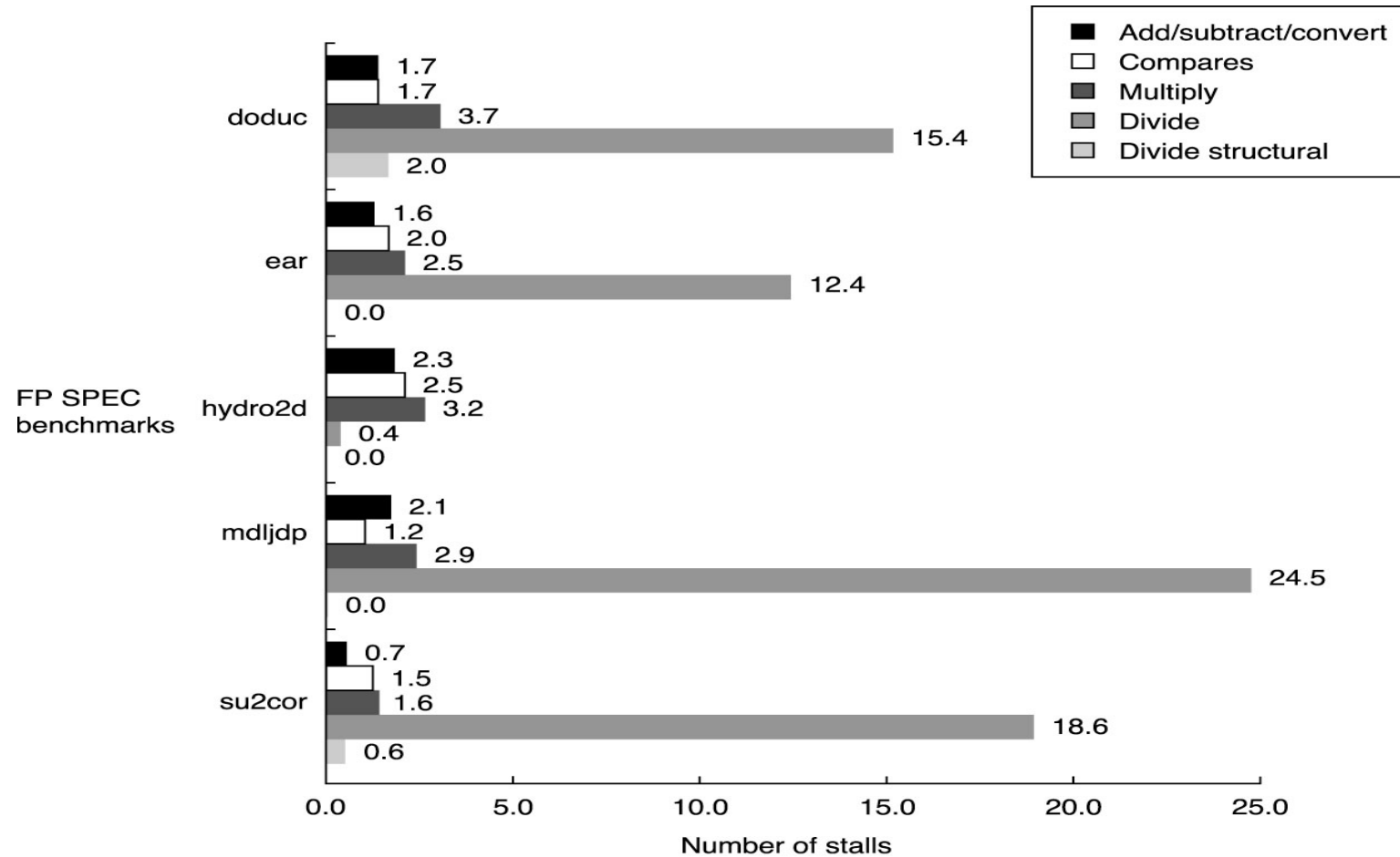
- Ignorar o problema (interrupções terão que ser imprecisas) – 1960's e 1970's [supercomputadores]
  - Memória virtual e IEEE FP fazem esta opção ser difícil de implementar (requerem interrupções precisas)
  - Dois modos de operação para alguns processadores (DEC Alpha e MIPS R8000)
    - lento para interrupções precisas
    - rápido para interrupções imprecisas
- Guardar resultados até operações iniciadas antes da operação que completou fora de ordem poderem completar (forwarding?)
  - History file - CYBER 180/990
  - Future file - PPC, MIPS R10k

# Solução para Manter Interrupções Precisas

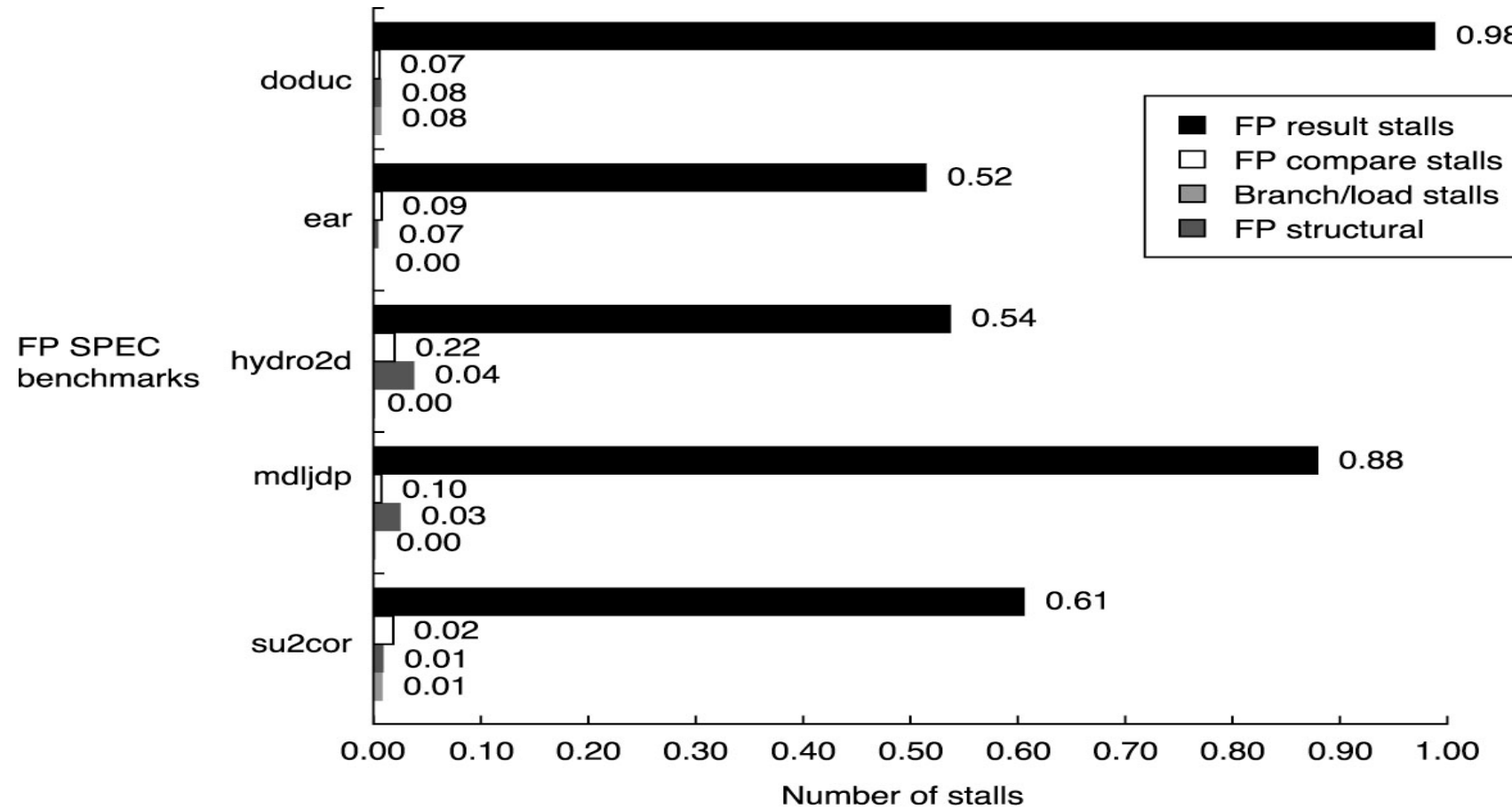
---

- Interrupções imprecisas com informações para recuperação por software - SPARC
  - Precisa saber quais os PCs das instruções que estão no pipe e quais instruções terminaram
  - Instrs (1,2,3,...,n)
    - 1 é longa
    - n completou
- Permite nova instrução ser executada somente se instruções anteriores estão certas de terminarem sem erros
  - Usado no Pentium, MIPS R2000, R3000, R4000

# Desempenho do MIPS FP



# Stalls Ocorrendo no MIPS

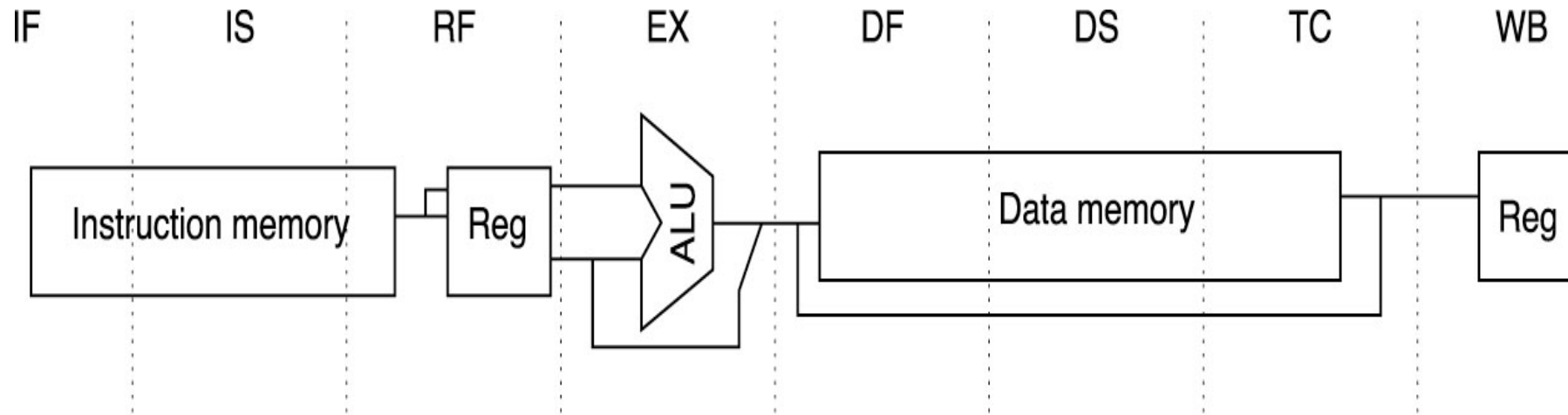


**MIPS 4000**



# Pipeline do MIPS 4000

---

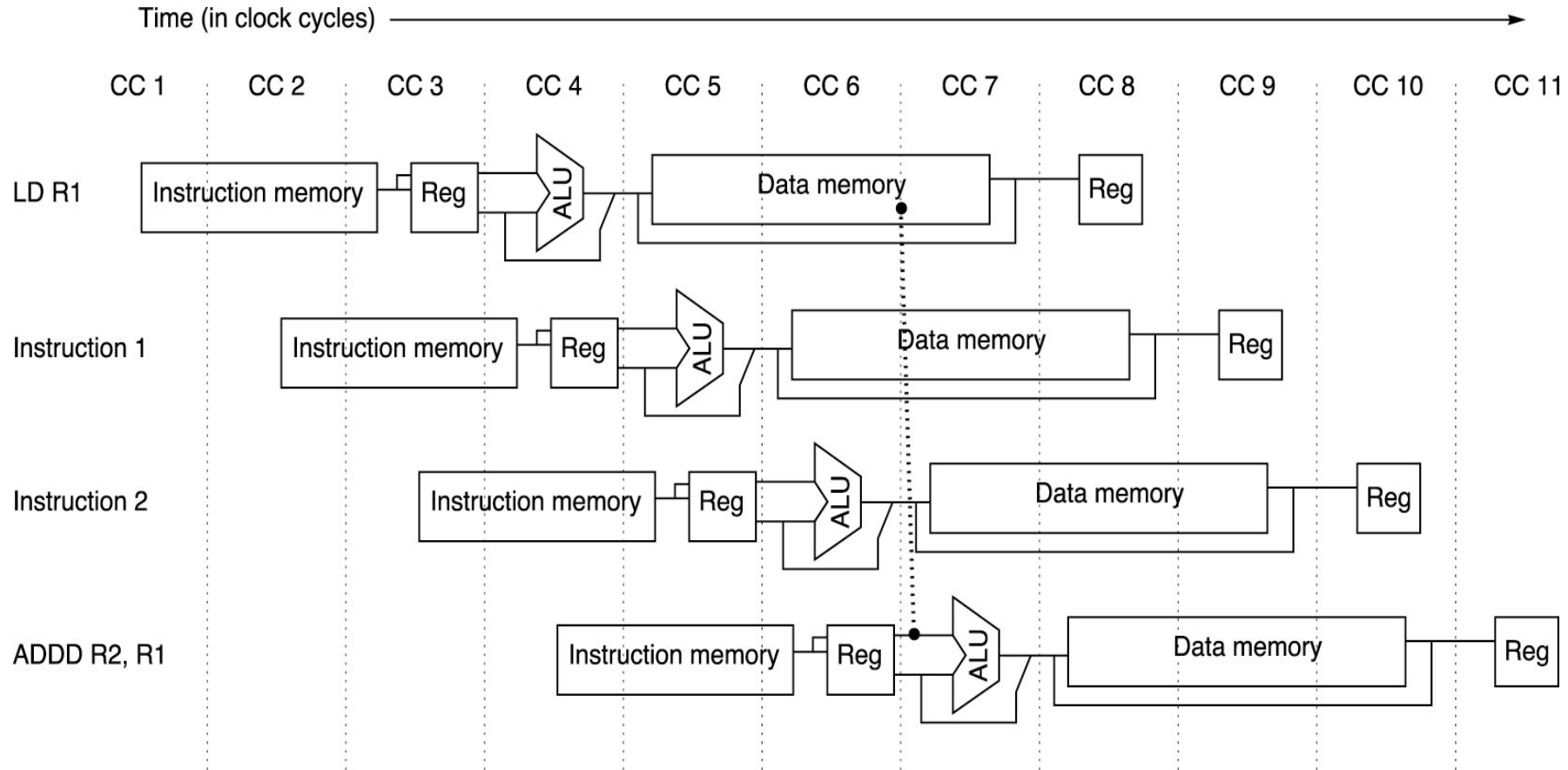


# MIPS R4000

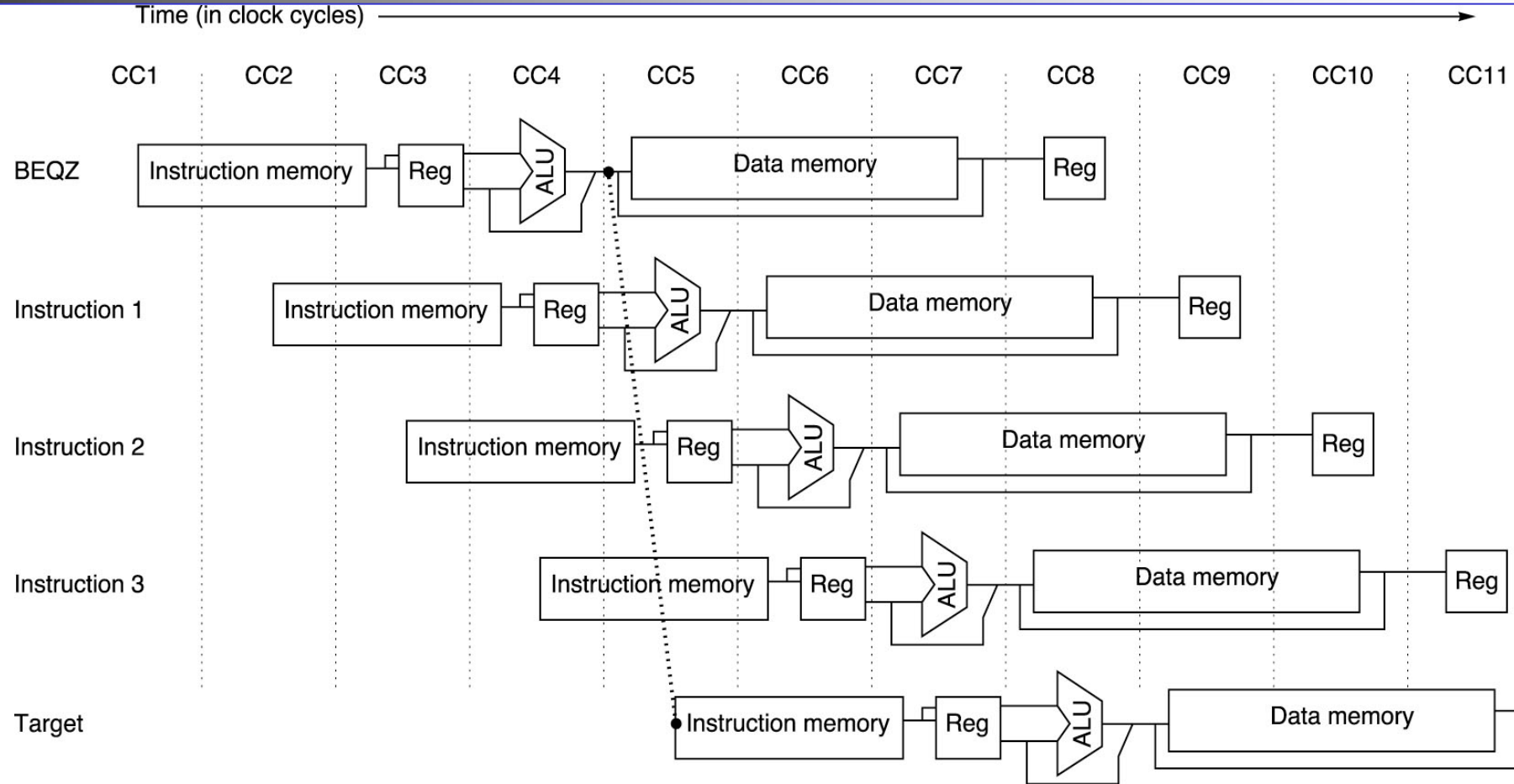
---

- Pipeline de 8 estágios:
  - **IF**—1/2 ciclo de busca da instrução; seleção de PC ocorre neste ciclo assim como início do acesso à cache de instrução.
  - **IS**—2/2 ciclo de busca na cache de instrução.
  - **RF**—decodificação de instrução e busca de operandos em registradores, verificação de hazards e detecção de hit na cache de instruções.
  - **EX**—execução, que inclui cálculo do endereço efetivo, ALU e cálculo do endereço do target de um branch e avaliação da condição.
  - **DF**—busca de dados, 1/2 do acesso à cache de dados.
  - **DS**—2/2 ciclo de acesso à cache de dados.
  - **TC**—verificação de tags, detecção de hit na cache de dados.
  - **WB**—write back para loads e operações de ALU reg-reg.
- 8 Estágios: Qual o impacto no load delay e branch delay?

# Load Delay para MIPS 4000



# Branch Delay para MIPS 4000



# MIPS R4000

## DOIS ciclos para latência de loads

IF	IS	RF	EX	DF	DS	TC	WB
	IF	IS	RF	EX	DF	DS	TC
		IF	IS	RF	EX	DF	DS
			IF	IS	RF	EX	DF
				IF	IS	RF	EX
					IF	IS	RF
						IF	IS
							IF

## TRÊS ciclos para latência de branches

(condições avaliadas em EX)

**Um delay slot mais dois stalls**  
**Branch pode cancelar stalls se**  
**não for tomado (*predict not taken*)**

IF	IS	RF	EX	DF	DS	TC	WB
	IF	IS	RF	EX	DF	DS	TC
		IF	IS	RF	EX	DF	DS
			IF	IS	RF	EX	DF
				IF	IS	RF	EX
					IF	IS	RF
						IF	IS
							IF

Processor stalls  
 Processor stalls because  
 (conflict not taken)

# MIPS R4000 Floating Point

---

- FP Adder, FP Multiplier, FP Divider
- Último estágio de FP Multiplier/Divider usa hardware de FP Adder
- 8 tipos de estágios nas unidades de FP:

Stage	Functional unit	Description
A	FP adder	Mantissa ADD stage
D	FP divider	Divide pipeline stage
E	FP multiplier	Exception test stage
M	FP multiplier	First stage of multiplier
N	FP multiplier	Second stage of multiplier
R	FP adder	Rounding stage
S	FP adder	Operand shift stage
U		Unpack FP numbers

# Estágios no Pipe de FP do MIPS

---

<i>FP Instr</i>	1	2	3	4	5	6	7	8	...
Add, Subtract	U	S+A	A+R	R+S					
Multiply	U	E+M	M	M	M	N	N+A	R	
Divide	U	A	R	D <sup>27</sup>	...	D+A	D+R, D+A, D+R, A, R		
Square root	U	E	(A+R) <sup>108</sup>	...		A	R		
Negate	U	S							
Absolute value	U	S							
FP compare	U	A	R						

*Stages:*

*M*      *First stage of multiplier*  
*N*      *Second stage of multiplier*  
*R*      *Rounding stage*  
*S*      *Operand shift stage*  
*U*      *Unpack FP numbers*

*A*      *Mantissa ADD stage*  
*D*      *Divide pipeline stage*  
*E*      *Exception test stage*

# Desempenho do R4000

---

- Não é CPI Ideal = 1:
  - **Load stalls** (1 ou 2 ciclos)
  - **Branch stalls** (2 ciclos + slots não preenchidos)
  - **FP result stalls**: Hazard de dados do tipo RAW
  - **FP structural stalls**: Não possui hardware suficiente



# Desempenho do MIPS 4000

