

Universidade de Brasília

Departamento de Ciência da Computação

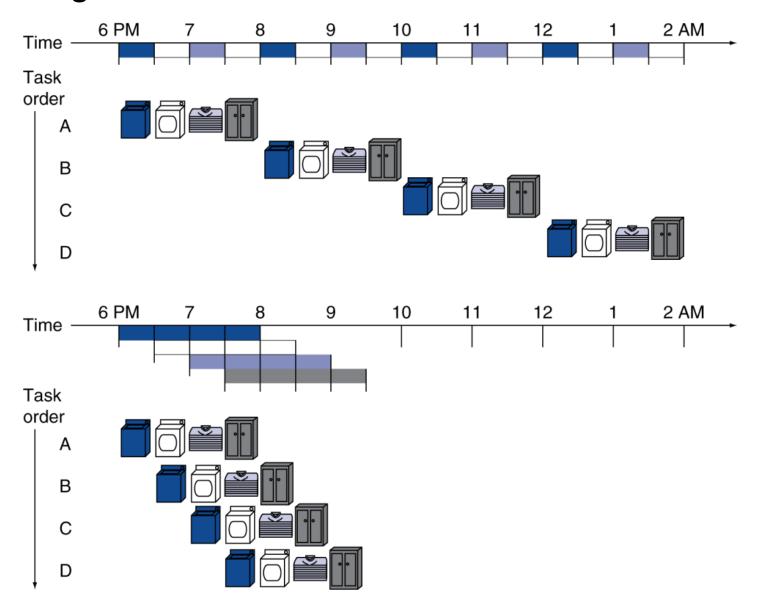
Aula 16 Implementação RISC-V Pipeline - Conceitos



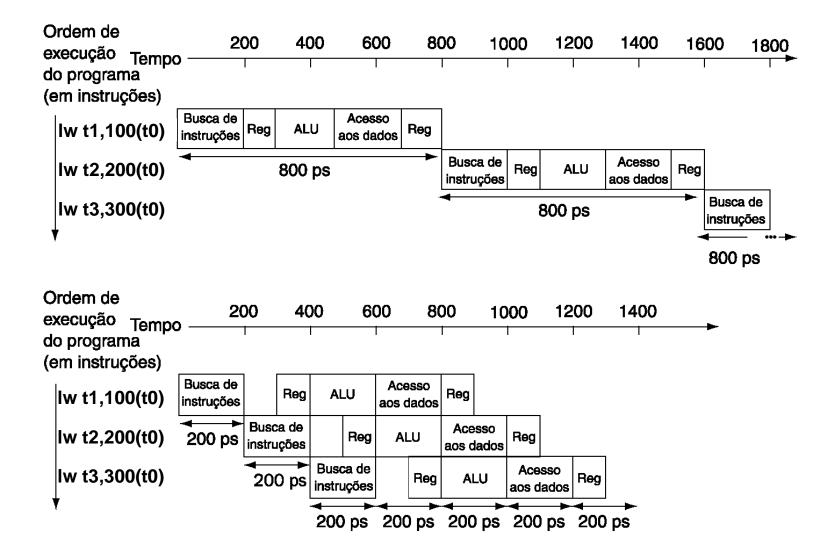


- Apresentar técnicas para aumento de desempenho de uma única tarefa em único processador usando ILP (Instruction Level Parallelism):
 - □ *Pipelining*: Técnica de projeto onde o *hardware* processa mais de uma instrução de cada vez, sem esperar que uma instrução termine antes de começar a próxima.
 - Superpipeline : Técnicas para aumento da eficiência do pipeline.
 - □ **Superescalar**: Uso eficiente de múltiplas unidades funcionais.
- Outros níveis de paralelismo: Aumenta a eficiência em máquinas multitask.
 - ☐ Multithread : Várias tarefas independentes simultaneamente, aproveitando-se das "falhas" do pipeline e/ou de estrutura superescalar. (Intel: Hyper Thread cores lógicos)
 - ☐ Multicore: Vários núcleos(cores físicos) em 1 chip, compartilhando memórias caches.
 - ☐ Multiprocessor: Sistema multiprocessado/distribuído. Diferentes computadores.

Pipelining: Conceito Básico



Uniciclo vs Pipeline





- Latência: 5 ciclos. Vazão: 1 instrução/ciclo
- Qual a aceleração ideal?
 - □ Para estágios balanceados, condições ideais e grande número de instruções:

$$Tempo\ entre\ Instruções_{com\ pipeline} = \frac{Tempo_entre_Instruções_{sem\ pipeline}}{N\'umero\ de\ est\'agios\ do\ pipeline}$$

- □ Porque estágios balanceados? (sem lacunas)
- □ Por que condições ideais? (sem hazards)
- □ Porque grande número de instruções? (desconsiderando o efeito da latência)

Ex.: 3 instruções: 2400 vs 1400

1.000.000 instruções: 1.000.000x800 vs 999.999×200+1000

Pipeline

O que facilita o pipeline:

□ Todas instruções têm o mesmo comprimento.

Obs.: x86 possui instruções de 1 a 17 bytes, dificulta o cálculo do endereço da próxima instrução e a leitura da própria instrução da memória.

□ Poucos formatos de instruções.

Obs.: Os operandos (rs1,rs2,rd) estão sempre nas mesmas posições

□ Operandos em memória só aparecem em loads e stores.

Obs.: Pode-se usar a ULA para cálculo de endereço.

O mesmo não vale para busca de operandos da ULA da memória (x86-64)

O que complica (hazards):

- Riscos Estruturais
- □ Riscos de Dados
- □ Riscos de Controle



Hazard Estruturais

- Hardware pode não admitir a combinação de instruções em um mesmo ciclo de clock.
- Ou a unidade funcional está ocupada no momento.
- 1) O que aconteceria se só houvesse 1 pessoa na lavanderia?
- 2) Considerando apenas 1 memória para dados e instruções, como ficaria uma quarta instrução no exemplo dado?



- Pipeline precisa ser interrompido por que uma etapa precisa esperar até que outra seja concluída.
- O que fazer se 1 pé de meia seu foi lavado junto com as roupas de seu colega? (indivíduo com TOC)
- 2) Exemplo:

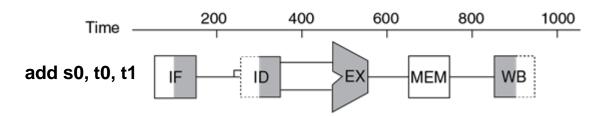
```
add s0, t0, t1 sub t2, s0, t3
```

Soluções:

- Inserção de Bolhas
- Execução fora de ordem (compilador e processador)
- Adiantamento, Forwarding ou Bypassing

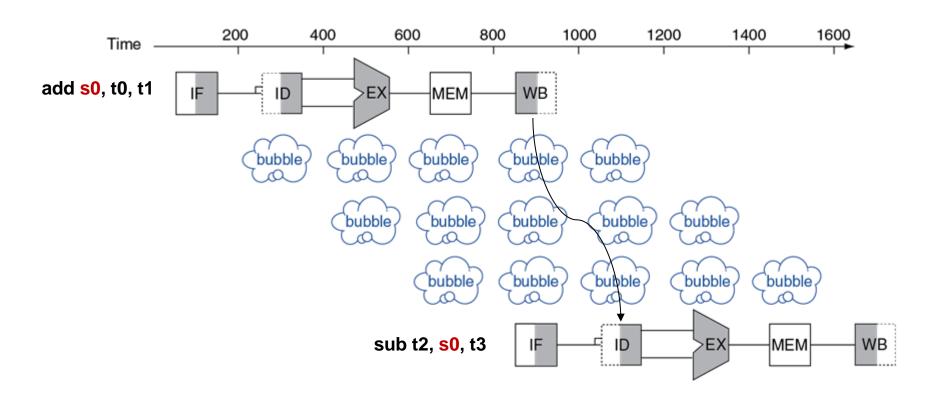
Inserção de Bolhas

Representação gráfica



Bolhas

□ Sabendo que o registrador s0 é escrito no 5º estágio



Inserção de Bolhas

Representação tradicional

Ciclo	01	02	03	04	05	06	07	80	09	10	11	12	13
add s0,t0,t1	IF	ID	EX	ME	WB \	\							
sub t2, <mark>s0</mark> ,t3		IF	ID*	ID*	ID*	ID	EX	ME	WB				
or s2,s3,s4			Æ*	IF*	IF*	IF	ID	EX	ME	WB			
slt t0,t1,t3		,					IF	ID	EX	ME	WB		

Aqui é identificado o hazard — 3 ciclos de **stall**

Parada do pipeline Injeta bolha



- Aproveitar os espaços das bolhas para tarefas úteis que sejam independentes.
- Alterar o algoritmo sem alterar o resultado!

□ Ex.:

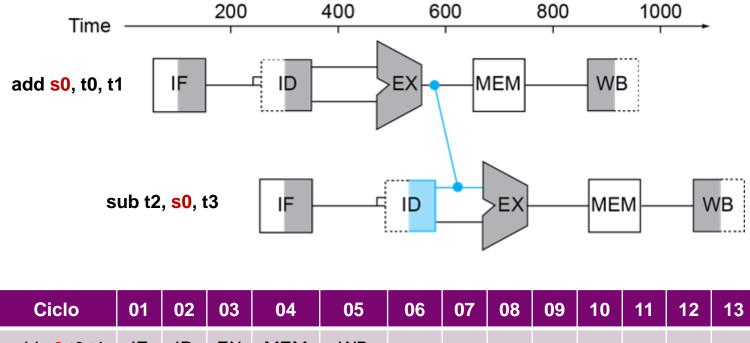
```
ori t6, t6, 100
add s1, zero, zero
addi t5, t4, 256
add s0, t0, t1
sub t2, s0, t3
```



add s0, t0, t1 ori t6, t6, 100 add s1, zero, zero addi t5, t4, 256 sub t2, s0, t3

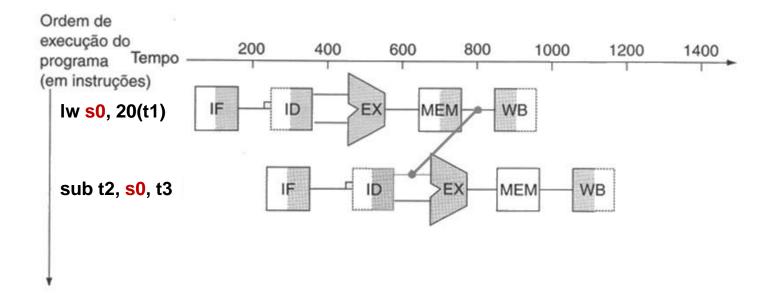
Forwarding

 Observação: não é necessário esperar que a instrução termine antes de tentar resolver o hazard de dados.



Forwarding

■ E se fosse um load?



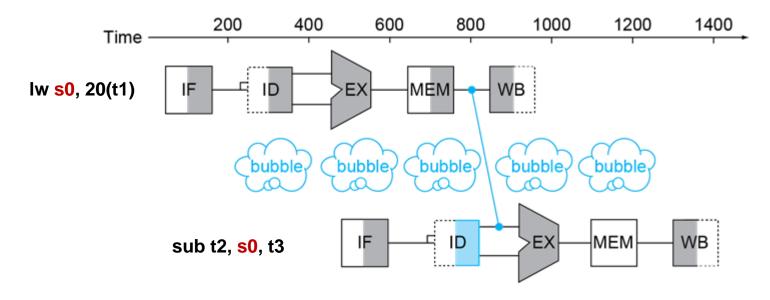
Em 600ps se necessita de um dado que estará disponível somente aos 800ps

Sistema que prevê o futuro!



Forwarding

■ Solução: Inserir uma bolha (stall) e dar forwarding



Ciclo	01	02	03	04	05	06	07	80	09	10	11	12	13
lw s0,20(t1)	IF	ID	EX	MEM \	WB								
sub t2, s 0,t3		IF	ID	ID*	EX	MEM	WB						
add s3,s2,s1			IF	IF*	ID	EX	MEM	WB					
or t0,t1,t3					IF	ID	EX	MEM	WB				



```
A=B+E; lw t1,0(fp)
lw t2,4(fp)
add t3,t1,t2
sw t3,12(fp)
lw t4,8(fp)
add t5,t1,t4
sw t5,16(fp)
```

Identifique os hazards.

Como solucionar utilizando apenas bolhas?

Quantos ciclos são necessários com latência?

Como solucionar de maneira mais eficiente?

Quantos ciclos são necessários sem latência?

Ideal
Identificando os hazards

	01	02	03	04	05	06	07	08	09	10	11
lw t1 ,0(fp)	IF	ID	EX	ME	WB						
lw t2 ,4(fp)		IF	ID	EX	ME	WB					
add t3,t1,t2			IF	ID	EX	ME	WB				
sw t3 ,12(fp)				IF	ID	EX	ME	WB			
lw t4 ,8(fp)					IF	ID	EX	ME	WB		
add t5 ,t1, t4						IF	ID	EX	ME	WB	
sw t5 ,16(fp)							IF	ID	EX	ME	WB

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
lw t1,0(fp)	IF	ID	EX	ME	WB																		
lw t2,4(fp)		IF	ID	EX	ME	WB																	
nop			*	*	*	*	*																
nop				*	*	*	*	*															
nop					*	*	*	*	*														
add t3,t1,t2						IF	† ID	EX	ME	WB													
nop							*	*	*	*	*												
nop								*	*	*	*	*											
nop									*	*	*	*	*										
sw t3,12(fp)										IF	ID	EX	ME	WB									
lw t4,8(fp)											IF	ID	EX	ME	WB								
nop												*	*	*	*	*							
nop													*	*	*	*	*						
nop														*	*	*	*	*					
add t5,t1,t4															IF	ID	EX	ME	WB				
nop																*	*	*	*	*			
nop																	*	*	*	*	*		
nop																		*	*	*	*	*	
sw t5,16(fp)																			IF	ID	EX	ME	WB

Na realidade o que ocorre:

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
lw t1,0(fp)	IF	ID	EX	ME	WB																		
lw t2,4(fp)		IF	ID	EX	ME	WB																	
add t3,t1,t2			IF	ID*	ID*	ID*	ID	EX	ME	WB													
sw t3,12(fp)				IF*	IF*	IF*	IF	ID*	ID*	ID*	₽ID	EX	ME	WB									
lw t4,8(fp)								IF*	IF*	IF*	IF	ID	EX	ME	WB								
add t5,t1,t4												IF	ID*	ID*	ID*`	ID	EX	ME	WB				
sw t5,16(fp)													IF*	IF*	IF*	IF	ID*	ID*	ID*	₽ID	EX	ME	WB

Operação com forwarding

	01	02	03	04	05	06	07	08	09	10	11	12	13
lw t1,0(fp)	IF	ID	EX	ME	Wβ								
lw t2,4(fp)		IF	ID	EX	MIE	WB							
nop			*	*	*	*	*						
add t3,t1,t2				IF	ID	₽EX	ME\	WB					
sw t3,12(fp)					IF	ID	EX	ME	WB				
lw t4,8(fp)						IF	ID	EX	ΜΕ	WB			
nop							*	*	*	*	*		
add t5,t1,t4								IF	ID	∱EΧ	ME\	WB	
sw t5,16(fp)									IF	ID	EX	ME	WB

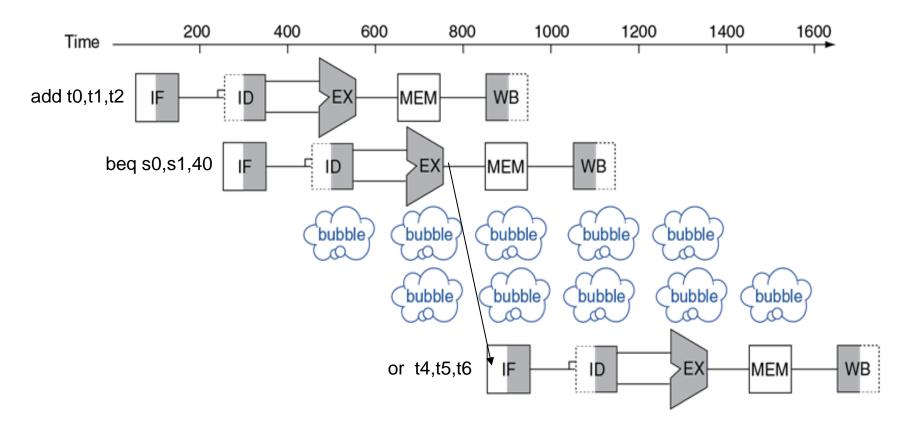
	01	02	03	04	05	06	07	08	09	10	11	12	13
lw t1,0(fp)	IF	ID	EX	ME	Wβ								
lw t2,4(fp)		IF	ID	EX	ME	WB							
add t3,t1,t2			IF	ID*	ıβ	EX	ME \	WB					
sw t3,12(fp)				IF*	IF	ID	EX	ME	WB				
lw t4,8(fp)						IF	ID	EX	MΕ\	WB			
add t5,t1,t4							IF	ID*	ID	₹ EX	MΕ\	WB	
sw t5,16(fp)								IF*	IF	ID	EX	ME	WB

Operação com forwarding e reordenamento

		01	02	03	04	05	06	07	08	09	10	11
	lw t1,0(fp)	IF	ID	EX	ME	WB						
	lw t2,4(fp)		IF	ID	EX	ME	WB					
, ▼	lw t4,8(fp)			IF	ID	EX	ME	WB				
	add t3,t1,t2				IF	ID	EX	ME \	WB			
	sw t3,12(fp)					IF	ID	EX	↓ ME	WB		
`	add t5,t1,t4						IF	ID	EX	ME \	WB	
	sw t5,16(fp)							IF	ID	EX	ME	WB

Hazard de Controle

 Necessidade de tomar uma decisão com base nos resultados de uma instrução enquanto outras estão sendo executadas.



BEQ é avaliado pela ULA no 3º estágio => Necessita 2 bolhas!!!

Hazard de Controle

Ciclo

add t0,t1,t2

beq s0,s1,16

and s4,s5,s6

ori t3,t4,255

or t5,t6,t7

02

ID

IF

01

IF

03

ID

IF

Na realidade:

End.

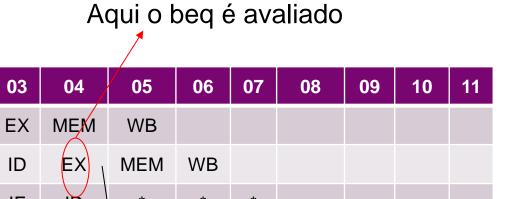
36

40

44

48

72



BEQ é avaliado pela ULA no 3º estágio => Necessita 2 Flushes !!!

IF

* IF

ID

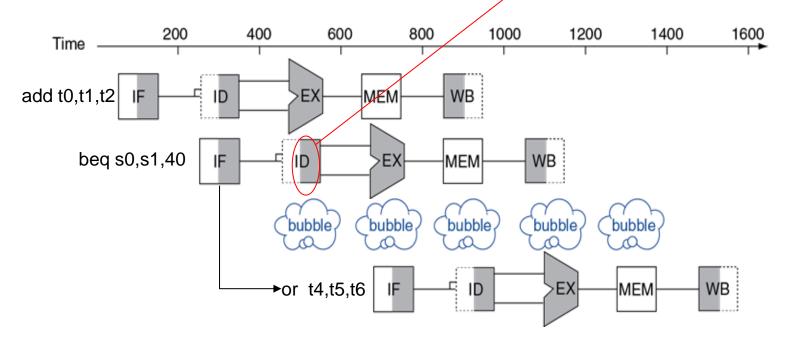
EX

Tornar a instrução um nop

MEM WB

Hazard de Controle

 Alterando o caminho de dados para que o BEQ seja avaliado no 2º Estágio => Economiza 1 Bolha!!



 Delayed Branch: ocupar a bolha com uma instrução útil, equivalente a atrasar a execução do Branch

No exemplo: add t0,t1,t2 no lugar da bolha : NÃO USADO NO RISC-V!

1400

1600

Hazard de Controle: Previsão de Desvios

Prevendo estaticamente o desvio como não tomado

400

600

800

1000

1200

200

Caso de: Desvio não tomado Acerto! "Vida que segue!"

Caso de:

Erro!

Desvio tomado

na instrução seguinte

add t0,t1,t2 WB MEM beq s0,s1,40 WB MEM lw s3,300(t3)WB 1200 200 400 600 800 1000 1400 1600 Time add t0,t1,t2 WB MEM Necessita dar um *flush* beq s0,s1,40 WB MEM lw s3,300(t3) (bubble) bubble bubble bubble bubble →or t4,t5,t6 WB MEM



Prevendo Quase-Dinamicamente o desvio

Grande parte das vezes os desvios são usados em loops:

```
LOOP: beq t0,t1,FIM

...

j LOOP

beq t0,t1, LOOP

FIM: ...
```

No RISC-V o Label é calculado como PC+Imm, assim se imm≥0 imm[31]=0 endereço à frente ⇒ Prevê não tomar o desvio se imm<0 imm[31]=1 endereço para trás ⇒ Prevê tomar o desvio

Hazard de Controle: Previsão de Desvios

Prevendo Dinamicamente o desvio

1 Bit

 □ Considera que a previsão é a mesma da última escolha

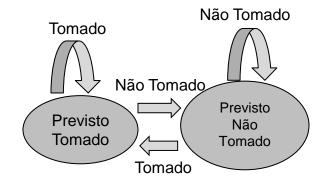
■ 2 Bits

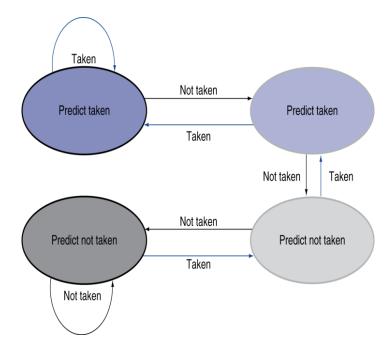
Considera uma variação mais conservativa.
 MEF ao lado

Tabela de histórico

 Considera uma tabela com contador do histórico do número de vezes que o branch foi tomado e não tomado.

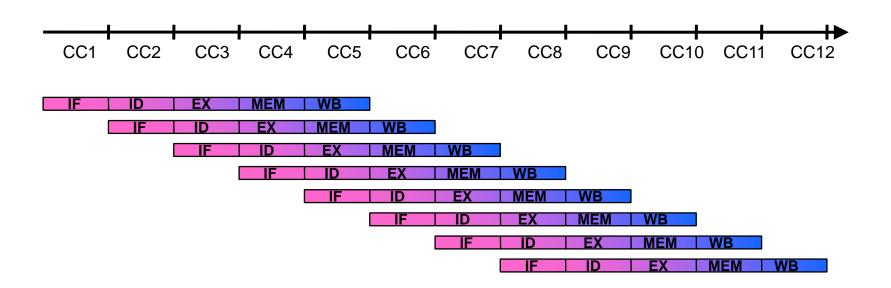
Mostrar Tool / BHT Simulator do Rars





Acelerando máquinas com pipeline

A vazão máxima de uma máquina com pipeline é de uma instrução por ciclo de clock.

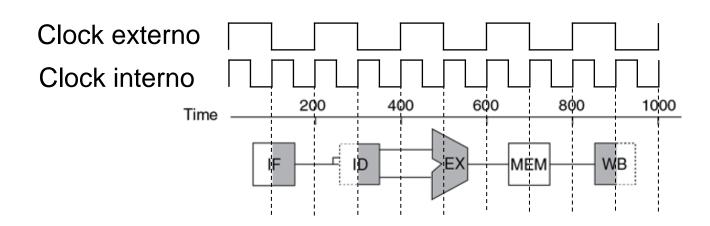


Técnicas para iniciar múltiplas instruções em 1 ciclo de clock:

- Estruturas superpipeline
- Estruturas superescalares.



- A estrutura Superpipeline subdivide cada estágio do pipeline em sub-estágios, e multiplica internamente a frequência de clock.
- Pipeline mais profundo → maior aceleração.
- Cada sub-estágio continua executando uma instrução por clock. Mas como o clock interno é multiplicado, o pipeline pode aceitar duas ou mais instruções para cada clock externo.
- Porém: Maior frequência → Maior dissipação térmica

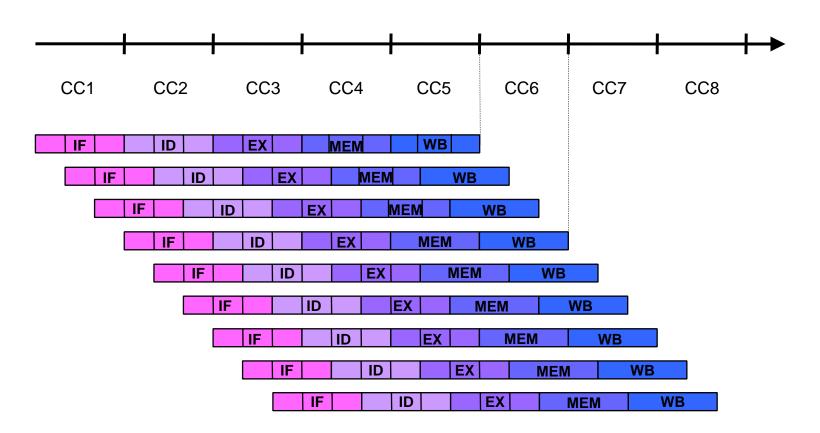


Superpipeline

Ex.: dividindo em 3 sub-estágios

CPI=?

Obs: Probabilidade de hazards aumenta → desempenho não é o esperado!

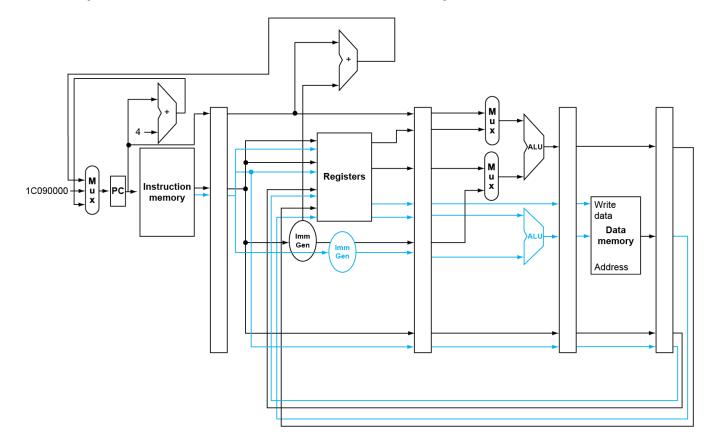




- A estrutura superescalar contém múltiplas unidades funcionais que são capazes de fazer a mesma tarefa.
- Isto permite ao processador executar várias instruções concorrentemente, pelo roteamento das instruções às unidades de execução disponíveis.

Ex.:

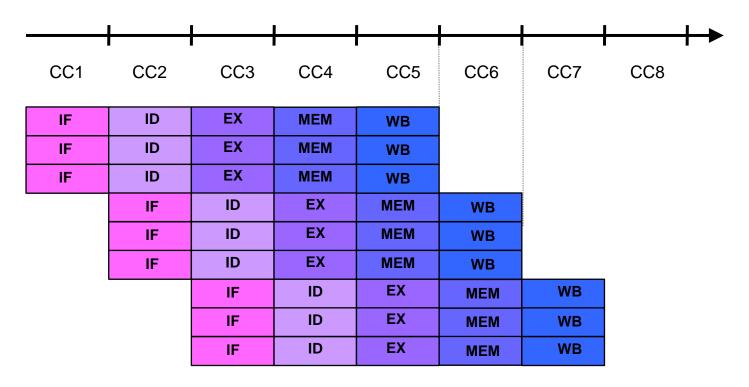
Tipo-R/I + Iw/sw



Pipeline Superescalar

Ex.: 3 réplicas da ULA CPI=?

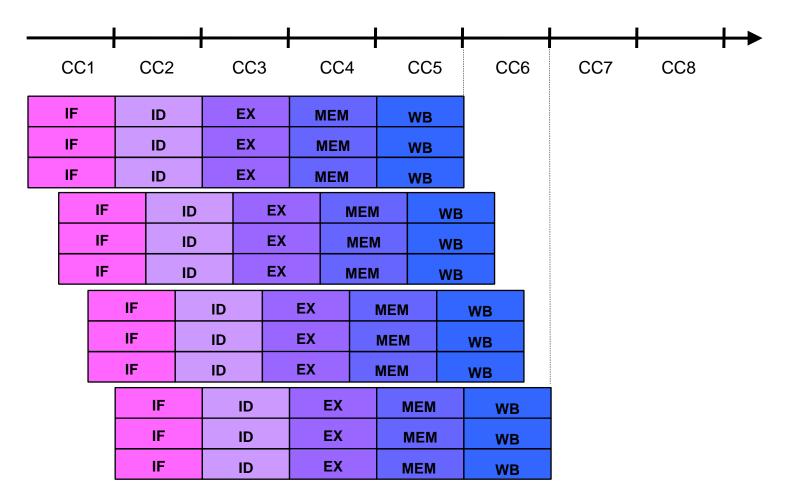
Obs.: Instruções independentes (sem hazards)



Hyper Threading (Processador Lógico): Duplicação do Banco de Registradores e aproveitamento eficiente das bolhas do pipeline.

→ Consome mais energia sem o ganho de desempenho esperado.





Quantas instruções são completadas a cada ciclo de clock externo?

IPC (Instruções por Ciclo) = ?