Explorando o paralelismo entre instruções

CES-25 – Arquiteturas para Alto Desmpenho
Prof. Paulo André Castro
pauloac@ita.br
Sala 110 – Prédio da Computação
www.comp.ita.br/~pauloac
IEC - ITA

Pipeline

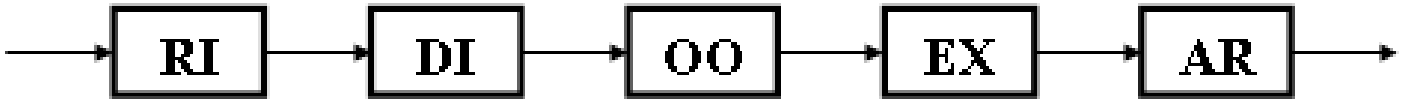
- Pipeline: Uma idéia natural
  - Linhas de montagem

- Dividir a tarefa em sub-tarefas seqüenciais, alocar recursos para cada sub-tarefa e controlar as mudanças de fase

- Ex: Lavanderia
  - Lavar e Secar: 4h. Logo, 4 cestas demorariam 16h
  - Lavar: 2h e Secar: 2h. Quanto 4 cestas demorariam?
  - 10h

Pipeline de Ciclo de Instrução

- **Pipeline do ciclo de instrução:** uma instrução pode ser (por exemplo) dividida em:
  - Recuperação da instrução (**RI**) IF, Decodificação da instrução (**DI**) ID, Obtenção dos operandos (**OO**) , Execução (**EX**) e Armazenamento do resultado (**AR**).



Instruções paralelizadas

- Sem pipeline, há muita **ociosidade e ineficiência**:

AR					I1					I2					I3	
EX				I1					I2					I3		
OO			I1					I2					I3			
DI		I1					I2					I3				
RI	I1					I2					I3					
					5					10					15	

- **Com pipeline**...

AR					I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	
EX				I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11		
OO			I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11			
DI		I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11				
RI	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11					
					5					10					15	

Qual o ganho de desempenho esperado com pipeline?

- Diagramas Estágio x Tempo

E																
E5					I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	
E4				I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11		
E3			I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11			
E2		I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11				
E1	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11					
					5					10					15	t

Conceitos Básicos

- **n**: número de instruções
- **p**: número de estágios do pipeline;
- **Tj** $(1 \leq j \leq p)$: demora em **Ej**;
- **TL**: demora de transição de estágio
- **T**: período de clock (ciclo de máquina)
- **Tmax= max{Ti}**
- **T = Tmax + TL**; **f**: freqüência = **1/T;**

Medindo o ganho...

- Gp: ganho com p estágios
  - Ganho = (Tempo sem pipeline)/(Tempo com pipeline)
  - Sem pipeline: t1 = n * p * T
  - Com pipeline: tp = (p + n – 1) * T
  - Gp = n*p / (p+n -1)
- Se n >> p, Gp aproximadamente p
- Então, quanto maior p maior o ganho ?

Eficiência do Pipeline

- η: eficiência – relação entre a **área ocupada** e a **área total** do diagrama Estágio x Tempo

$$\eta = \frac{n * p * T}{p * [(p-1) * T + n * T]} = \frac{n}{p + n - 1}$$

Quando $n \rightarrow \infty$, então $\eta = 1$ (máxima eficiência)

Produtividade (throughput)

- **W**: produtividade – número de tarefas completadas por unidade de tempo

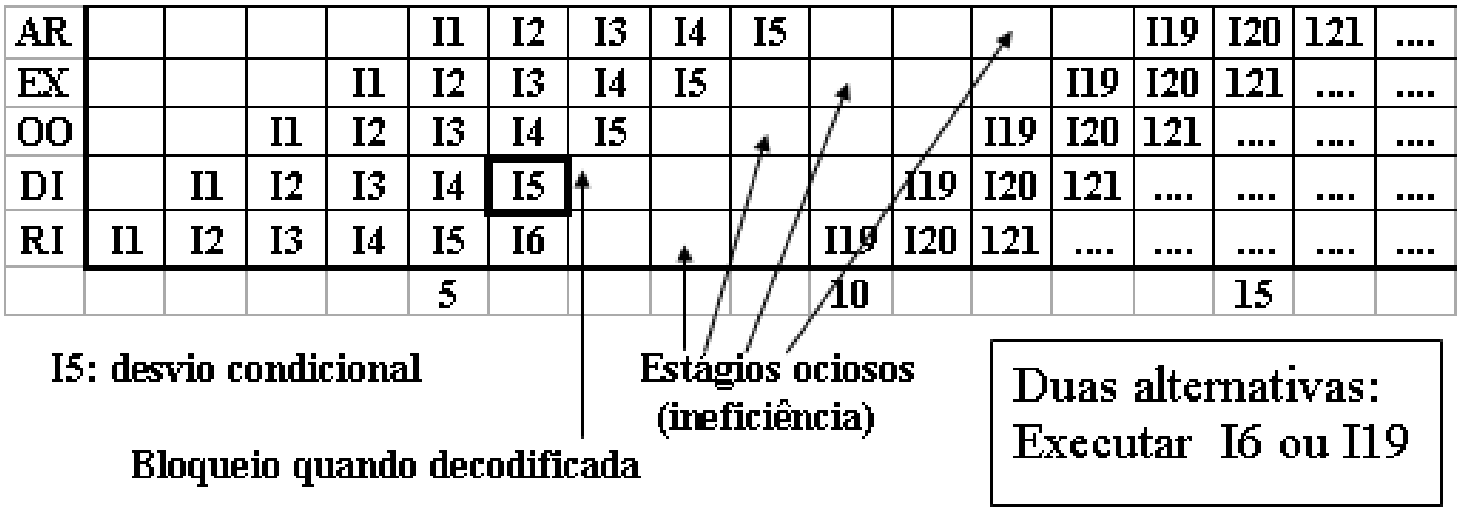
$$W = \frac{n}{(p + n - 1) * T} = \frac{\eta}{T}$$

Quando $\eta \rightarrow 1$, então $W \rightarrow 1/T = f$
(máxima produtividade)

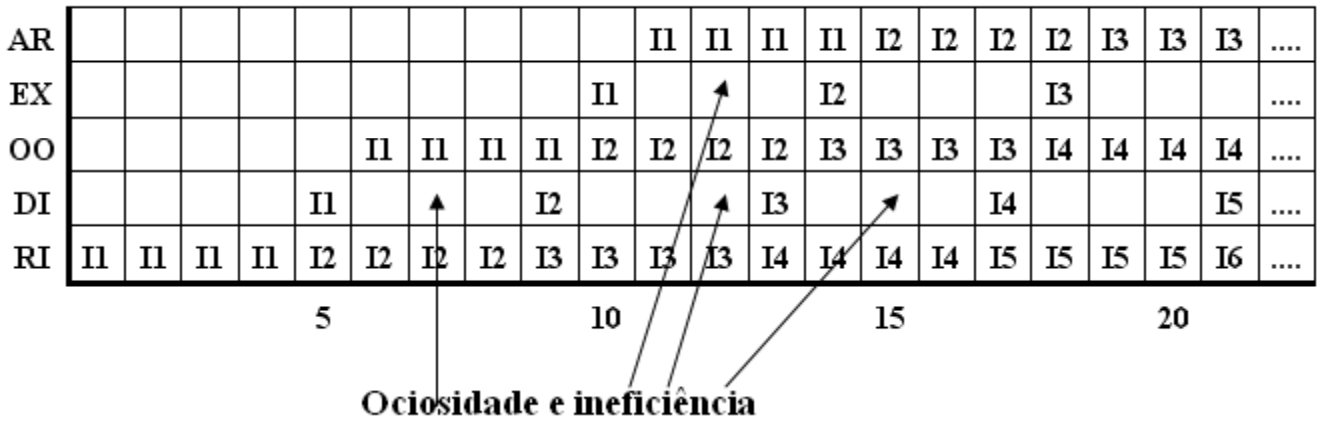
Impedimentos

- Impedimentos para **valores máximos** em ganho, eficiência e produtividade:
  - Estágios de diferentes tempos
    - Falhas em caches
    - Instruções de diferentes durações
  - Dependências
    - Recursos
    - Desvios e interrupções nos programas
    - Dados

Problemas no Pipeline



O Problema do Desnível entre Fases e Instruções



Problemas no Pipeline: Dependências

- Conflito de Recursos
- Dependência de Dados
  - RAW: Read After Write
  - WAR: Write After Read
  - WAW: Write After Write
- Dependência de Controle
  - Predição de Desvios, Desvios Retardados

Conflito de Recursos

- Também conhecido por Hazard Estrutural ou ainda Dependência Funcional
- Resultado da competição de duas ou mais instruções pelo mesmo recurso ao mesmo tempo
- Soluções:
  - uma das instruções deve esperar
  - aumento dos recursos

AR					I1	I2				I3	I4					
EX				I1	I2				I3	I4						
OO			I1	I2				I3	I4							
DI		I1	I2				I3	I4								
RI	I1	I2	I3	I3	I3	I3	I4	I5	I5	I5	I5	I6	I7			

Controle do Uso de Recursos

- Quando uma dependência envolve apenas **recursos da CPU** (registradores ou unidades funcionais), costuma-se usar uma **tabela de reserva** dos recursos.

- Toda instrução faz um **cheque na tabela** sobre os recursos que utilizará; se algum(uns) dele(s) estiver(em) marcado(s), ela é **bloqueada**, até eles sejam **liberados**.

- Após o uso, a instrução libera o recurso desmarcando a tabela de reserva.

Dependência de Dados

- Dependência WAR (Write After Read)
  - A = B+ C
  - B = C + D
  - Causa Problemas ?
- Falsa Dependência: sem problemas desde que não exista execução fora de ordem

Dependência de Dados

- Dependência WAW (Write After Write)
  - A = B+C
  - A = D+E
  - Causa Problemas ?
- Falsa Dependência: sem problemas desde que não exista execução fora de ordem

Dependência de Dados

- Dependência RAW (Read After Write)
  - I1: A = B+ C
  - I2: E = A + D
- Causa Problemas?
- Sim. Instrução I2 só pode captar o valor de A após o término da instrução I1
- Solução Simples
  - Atrasar instrução I2. Quantos Clocks?

Caminho dos Dados e Atrasos

CES-25 – Arquiteturas para Alto Desmpenho
Prof. Paulo André Castro
pauloac@ita.br
Sala 110 – Prédio da Computação
www.comp.ita.br/~pauloac
IEC - ITA

Pipeline MIPS

- O MIPS utiliza um pipeline com profundidade 5, porém com uma divisão de fases diferente da tradicional: RI, DI, OO, EX, AR. Fases do pipeline MIPS:
- IF: Instruction Fetch – carregamento de instrução
- ID/RF: Decodificação de registradores e carregamento de registradores
- EX: execução
- MEM: Acesso a memória para acessar dados (load e store)
- WB: Armazenamento do resultado

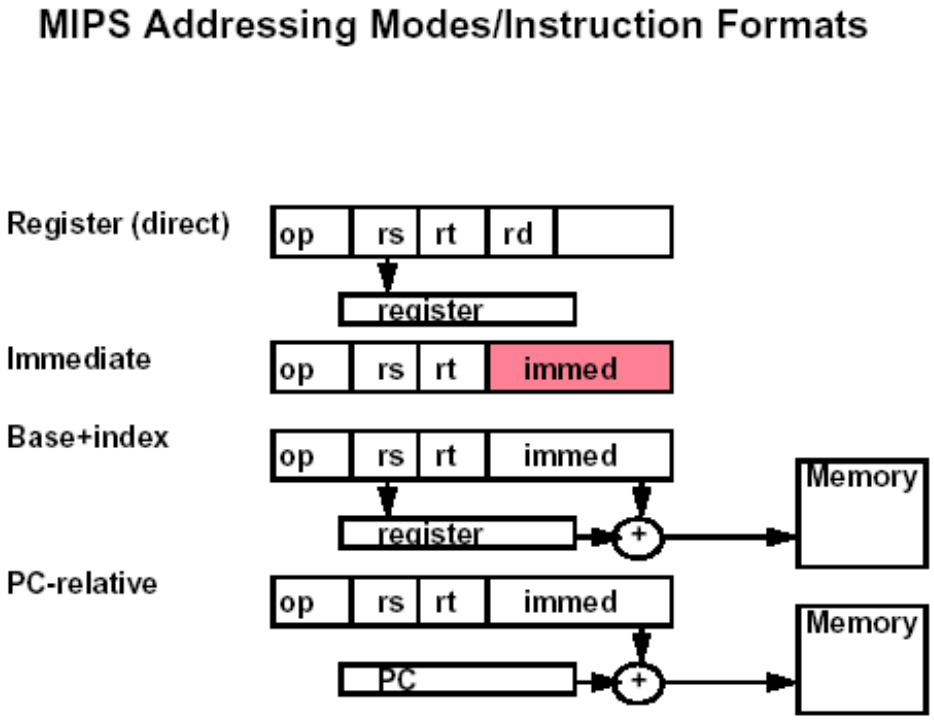
MIPS: Principais Instruções

- **Adição**
  - **add R1,R2,R3; R1 = R2 + R3**
- **Subtração**
  - **sub R1,R2,R3; R1 = R2 – R3**
- **Adição de constante (add immediate)**
  - **addi R1,R2,100; R1 = R2 + 100**
- **Multiplicação (resultado em 64 bits)**
  - **mult R2,R3; Hi, Lo = R2 x R3**
- **Divisão (resultado em 64 bits)**
  - **div R2,R3; Lo = R2 ÷ R3, Hi = R2 mod R3**
  - **Lo = quotient, Hi = remainder**

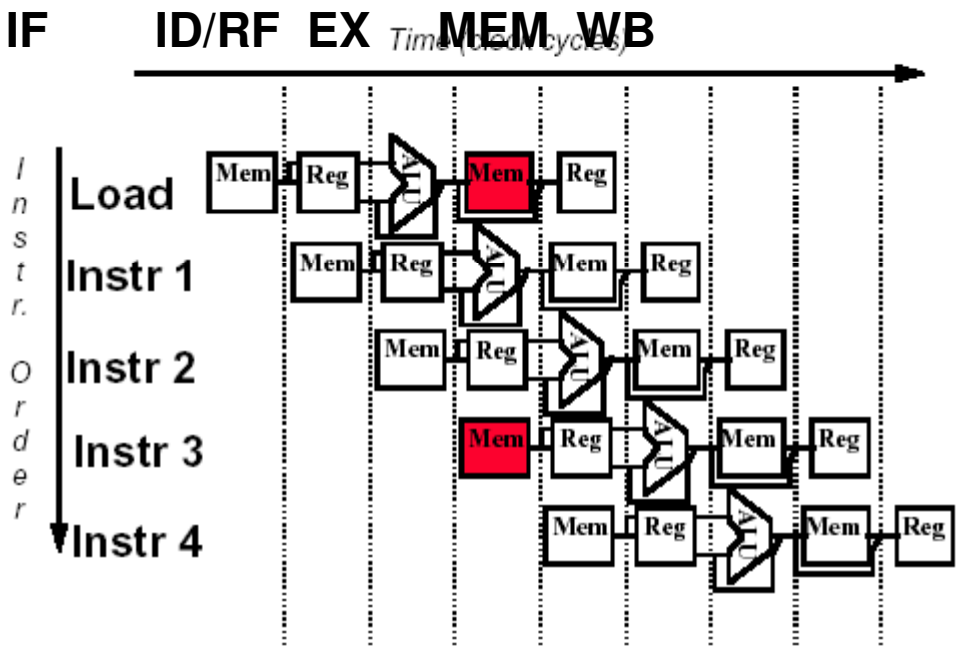
MIPS: Principais Instruções

- **Alterar memória (word)**
  - **SW R3, 500(R4) Mem[R4 + 500] =R3**
- **Ler memória (word)**
  - **LW R1, 30(R2) R1 = Mem[R2 + 30]**
- **Desvio Condicional**
  - **beq R1,R2,100 if (R1 == R2) go to PC+4+400**
- **Desvio incondicional (constante)**
  - **jump j 2500; go to 10000**
- **Desvio incondicional (registrador)**
  - **jr R31; go to R31**
- **Chamada de função (jump and link)**
  - **jal 2500 R31 = PC + 4; go to 10000**

Formato de Instruções MIPS



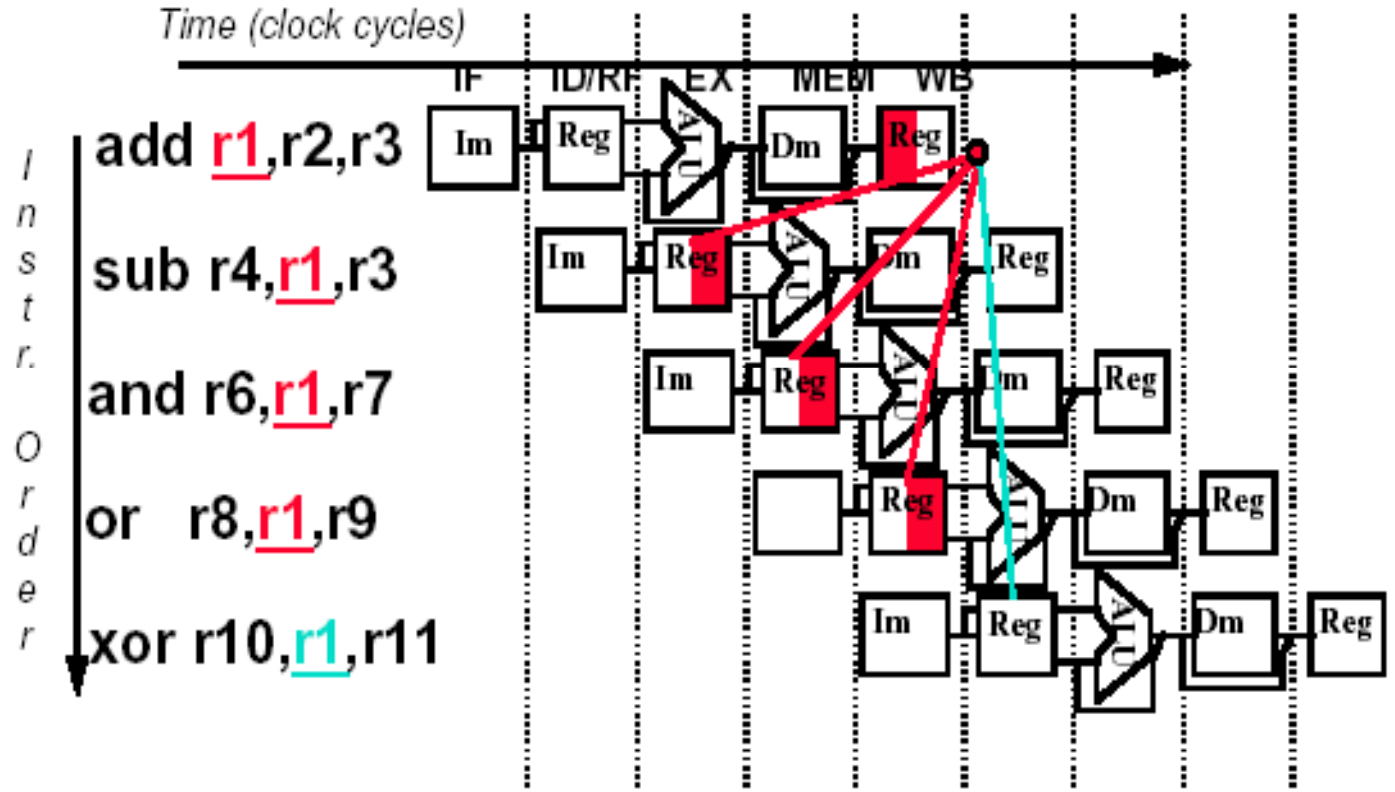
Visão do pipeline MIPS com Unidades Funcionais



Problemas no Pipeline: Dependências

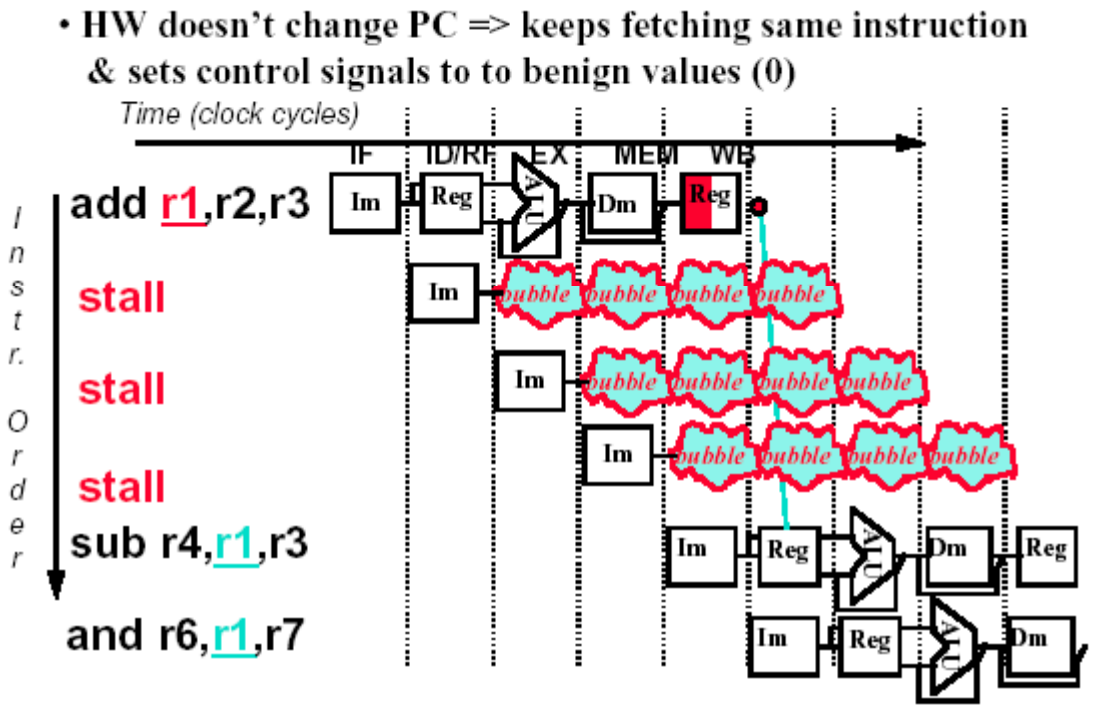
- Dependência (Conflito) de Recursos
- Dependência de Dados
  - RAW: Read After Write
  - WAR: Write After Read
  - WAW: Write After Write
- Dependência de Controle
  - Predição de Desvios, Desvios Retardados

Dependência de Dados - RAW

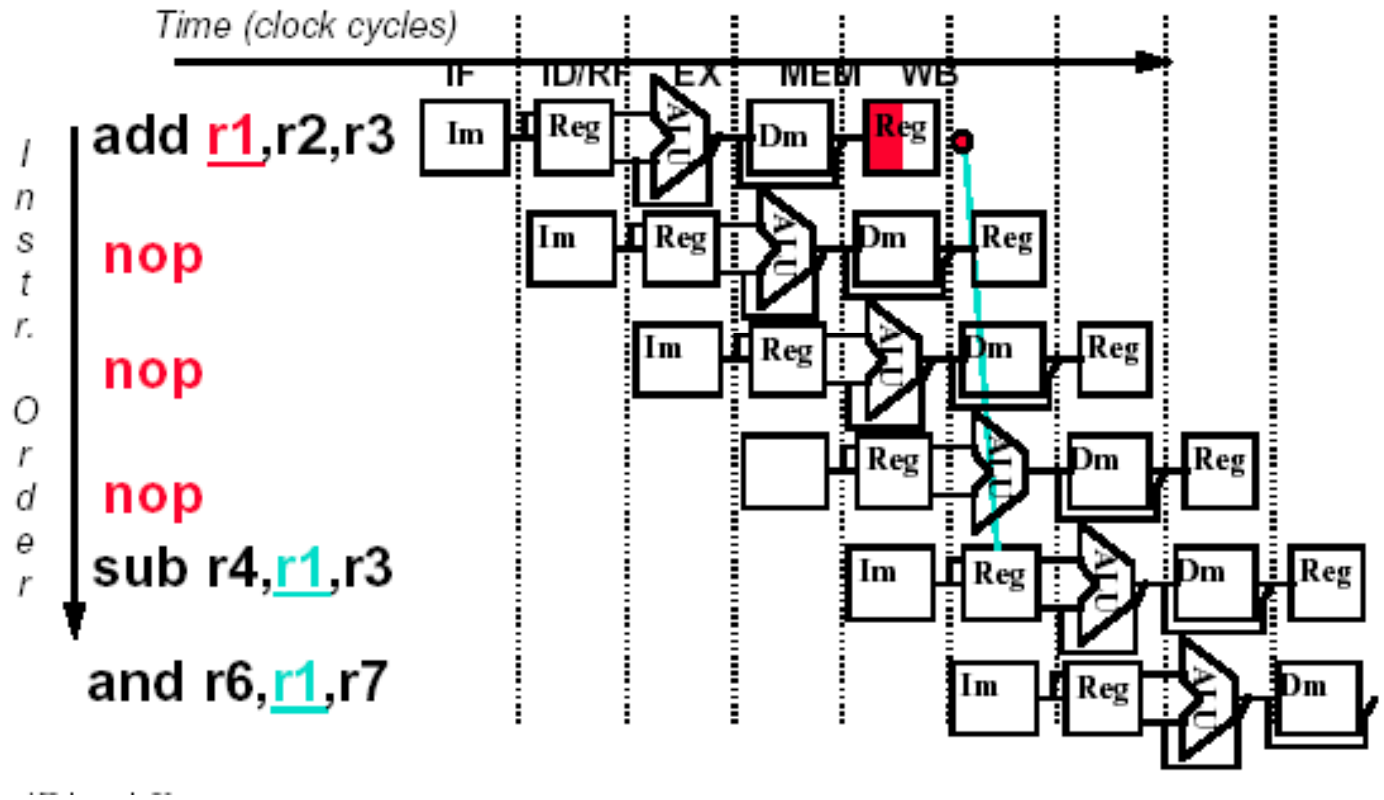


Solução: atrasar o pipeline (via SW ou via HW)

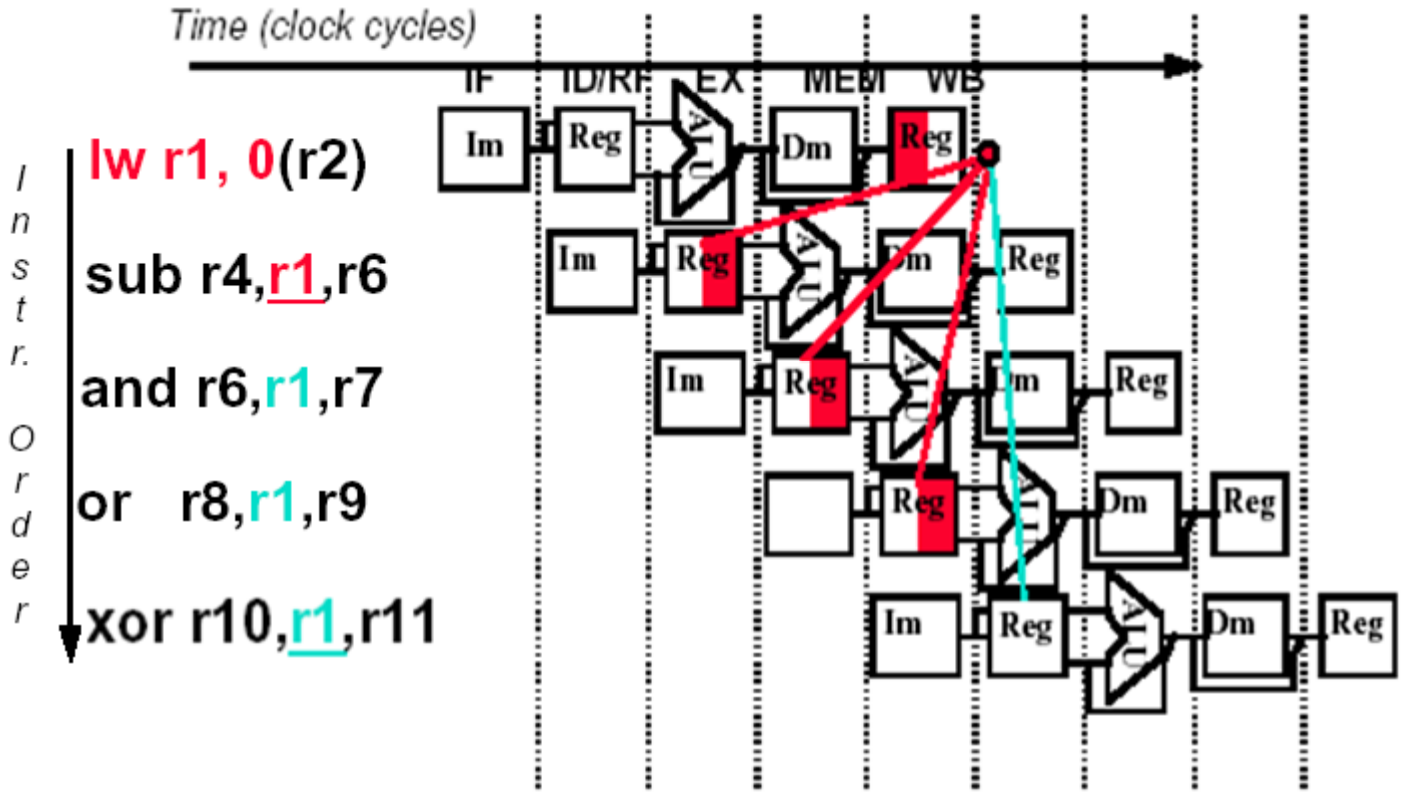
Implementando o atraso do Pipeline através de Hardware



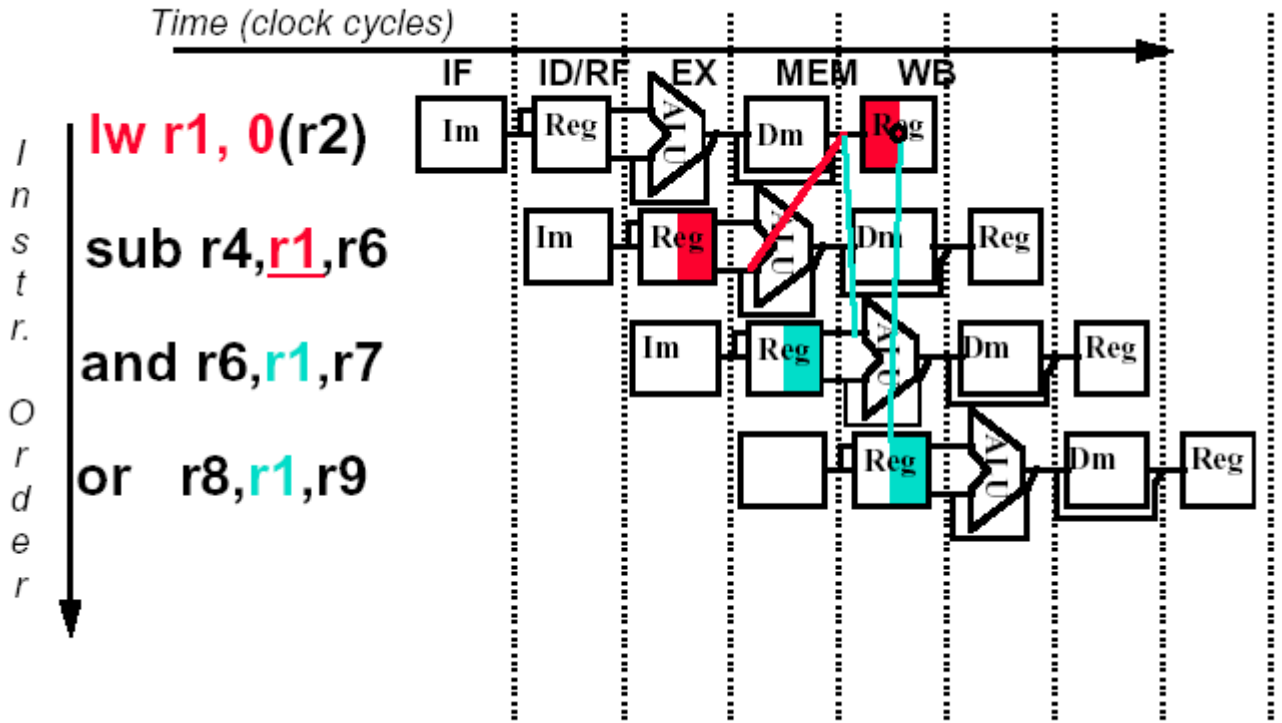
Implementando o atraso do pipeline através de software



Dependência RAW ao Carregar dados

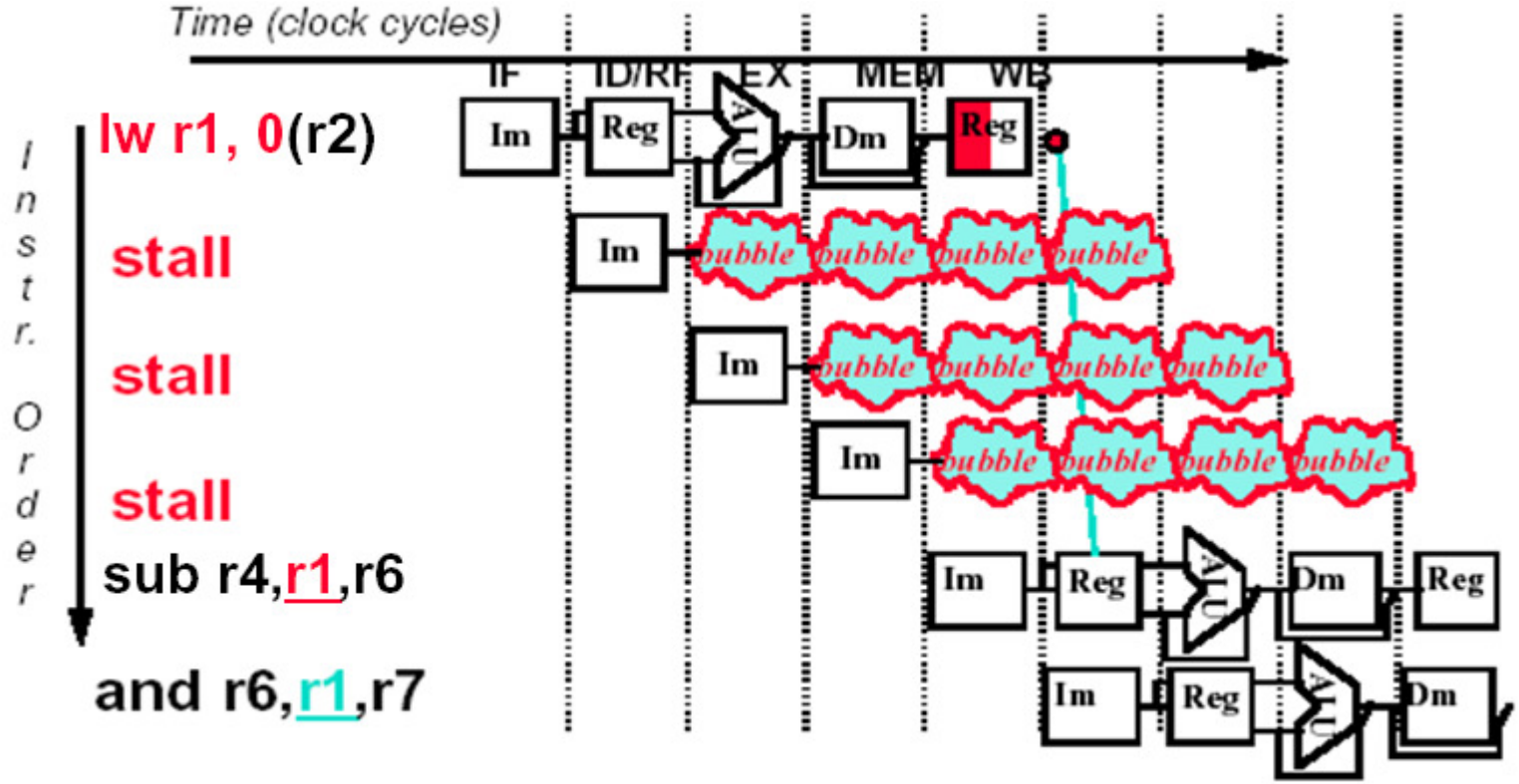


Dependência RAW no carregamento



Obs.: O dado estará disponível ao final de MEM, não ao final de EX como nas Operações R-type

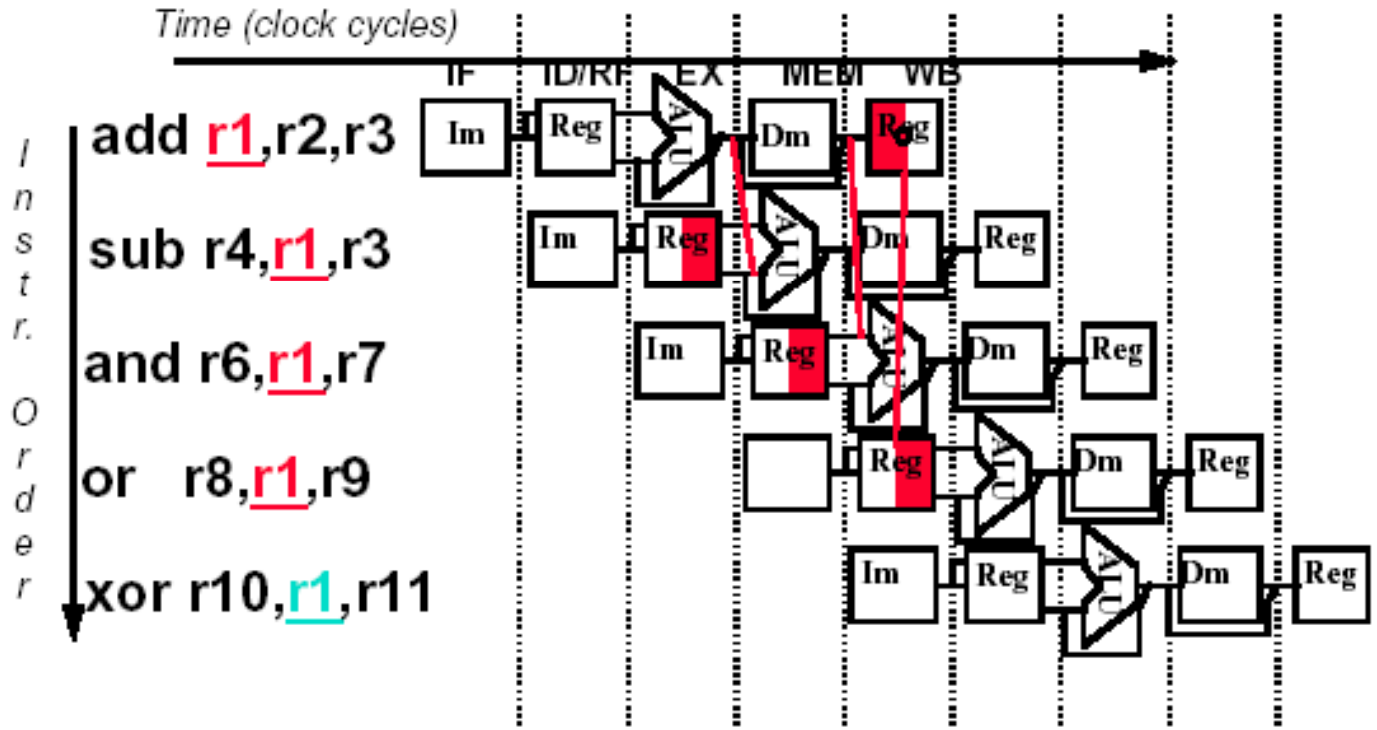
Atrasando o pipeline em carregamento de dados – Quantos clocks são necessários?



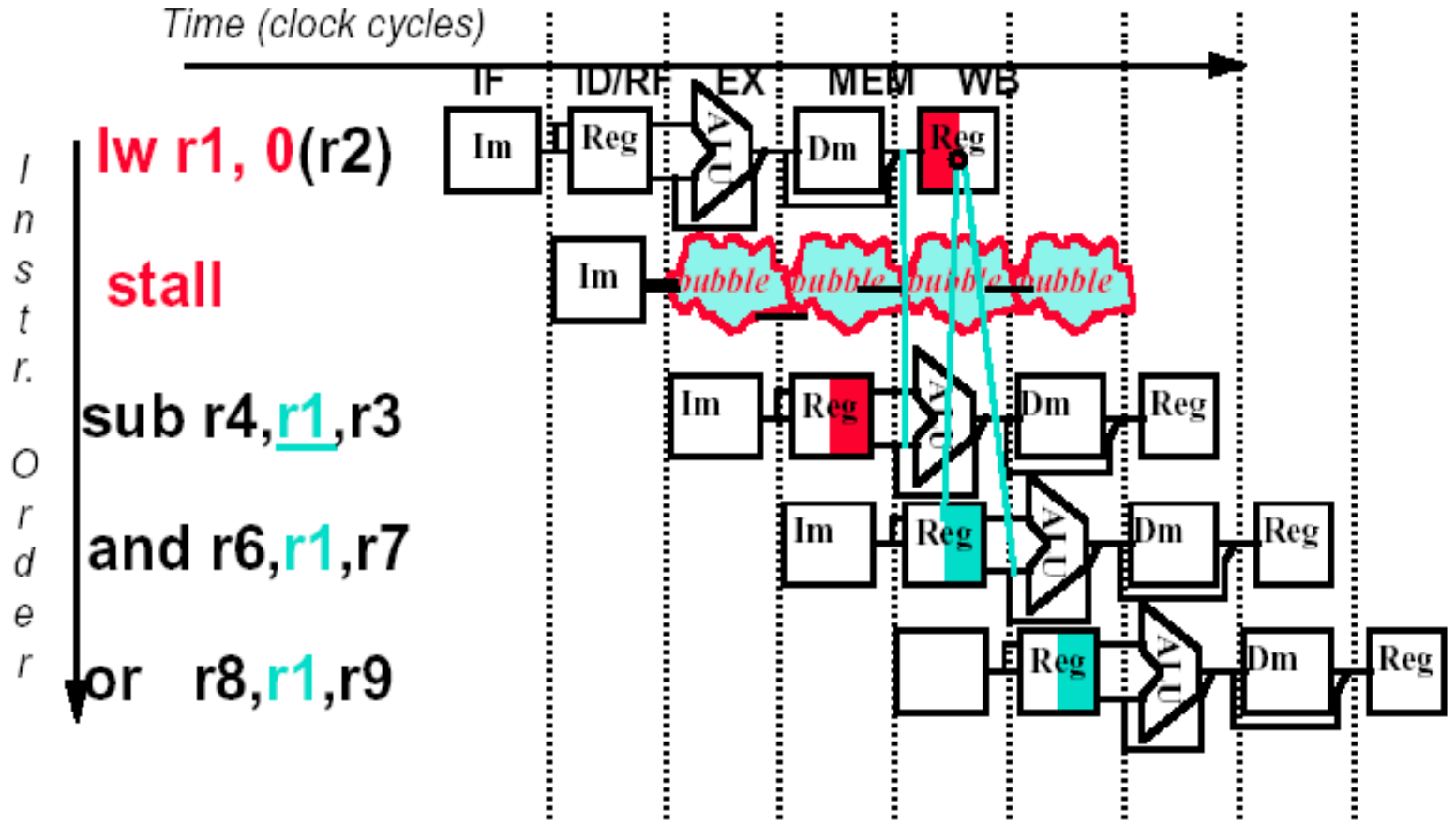
Dependências RAW

- Esses problemas são comuns?
  - Operações R-type: 3 bolhas (stalls) por dependência
  - Logo, 3 unidades funcionais ociosas.
- Há como evitar a ocorrência da dependência?
- Há como diminuir a ociosidade ?

Redução de Penalidade em R-Type: Bypassing

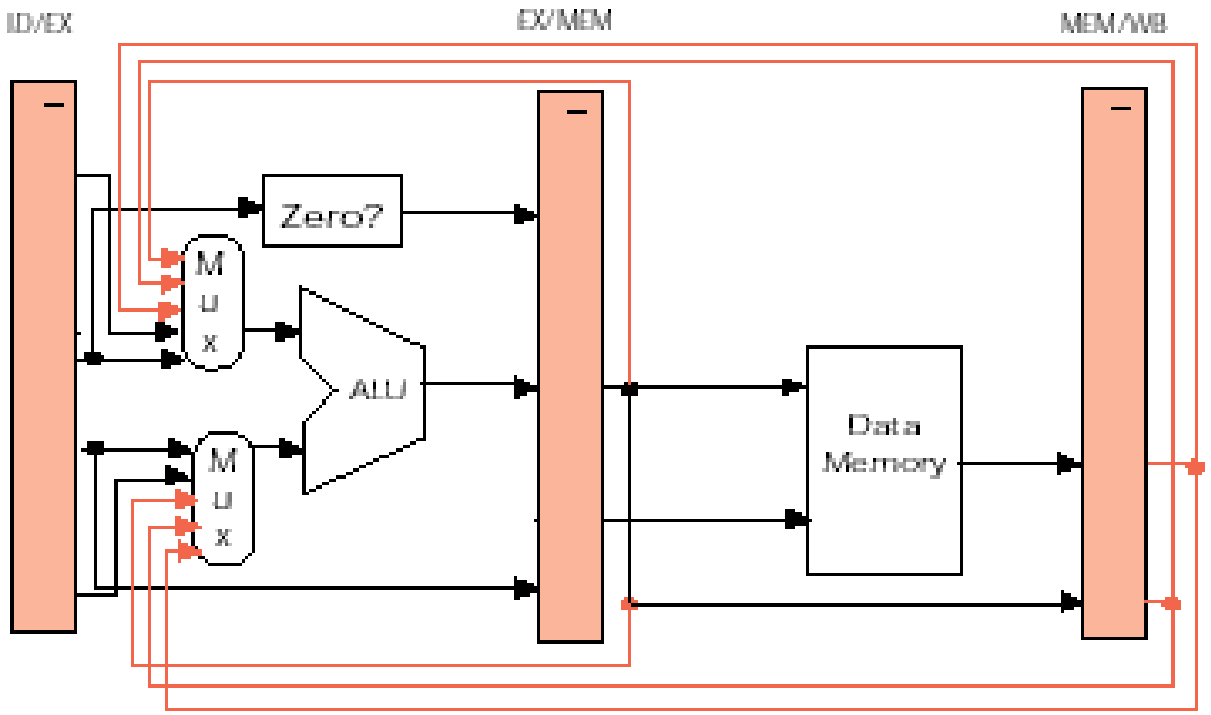


Encaminhamento em Carregamentos



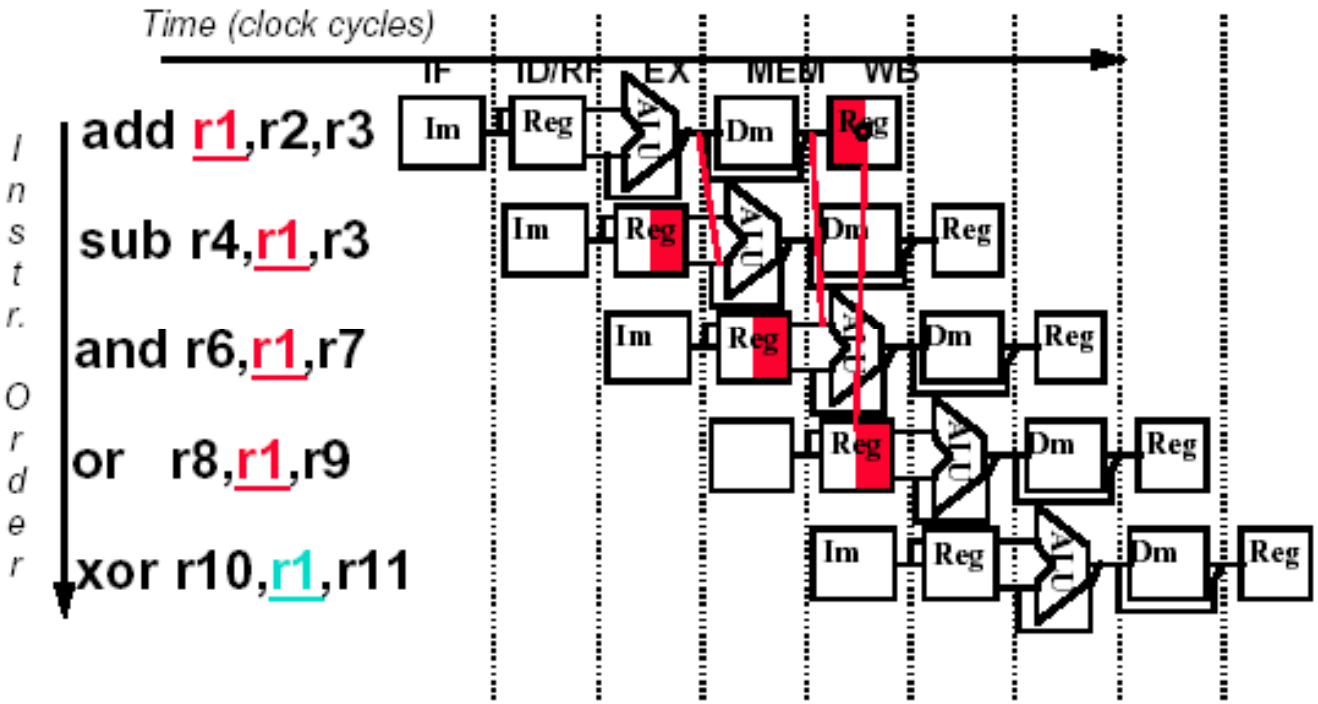
Seria possível eliminar este atraso ?

Implementando o Bypassing

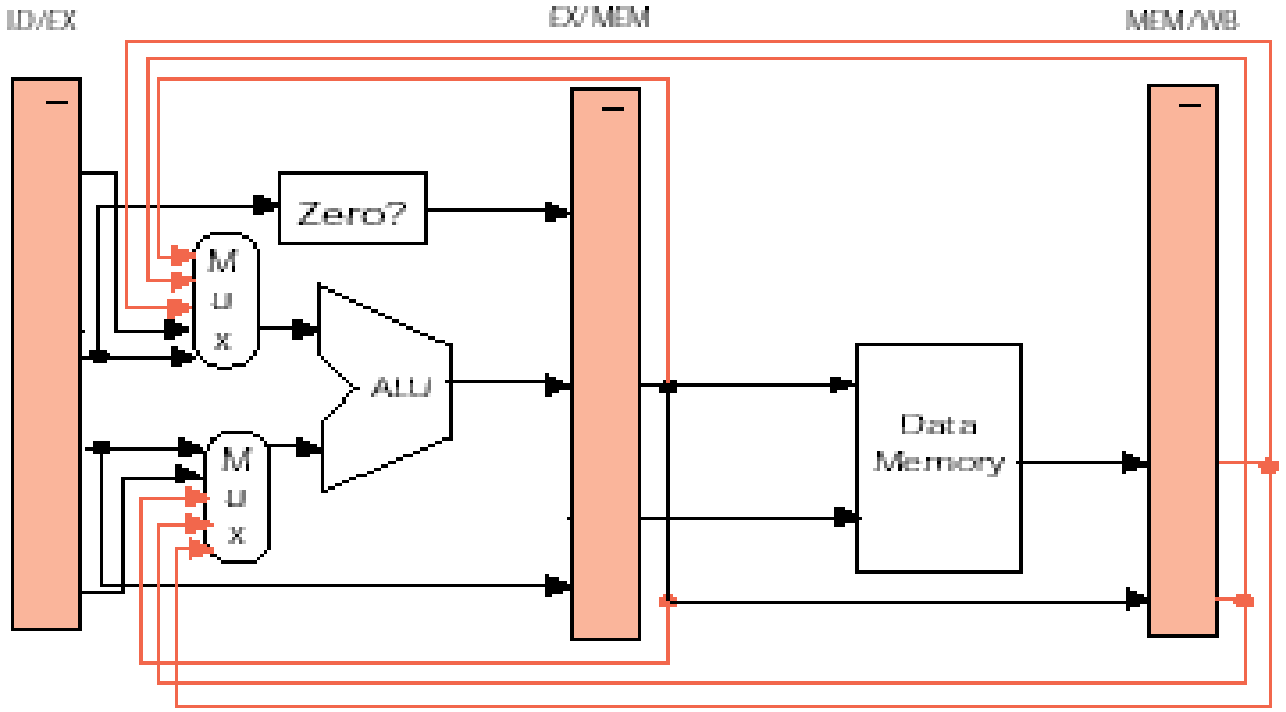


- Necessário multiplexadores com mais entradas para receber novos dados
- Assume-se que durante ID/RF, as gravações são feitas antes das leituras

Como controlar o Encaminhamento?



O problema do Encaminhamento



Com bypassing (novas opções): Saída da EX, Saída da DM (memória) e Saída da DM (EX)

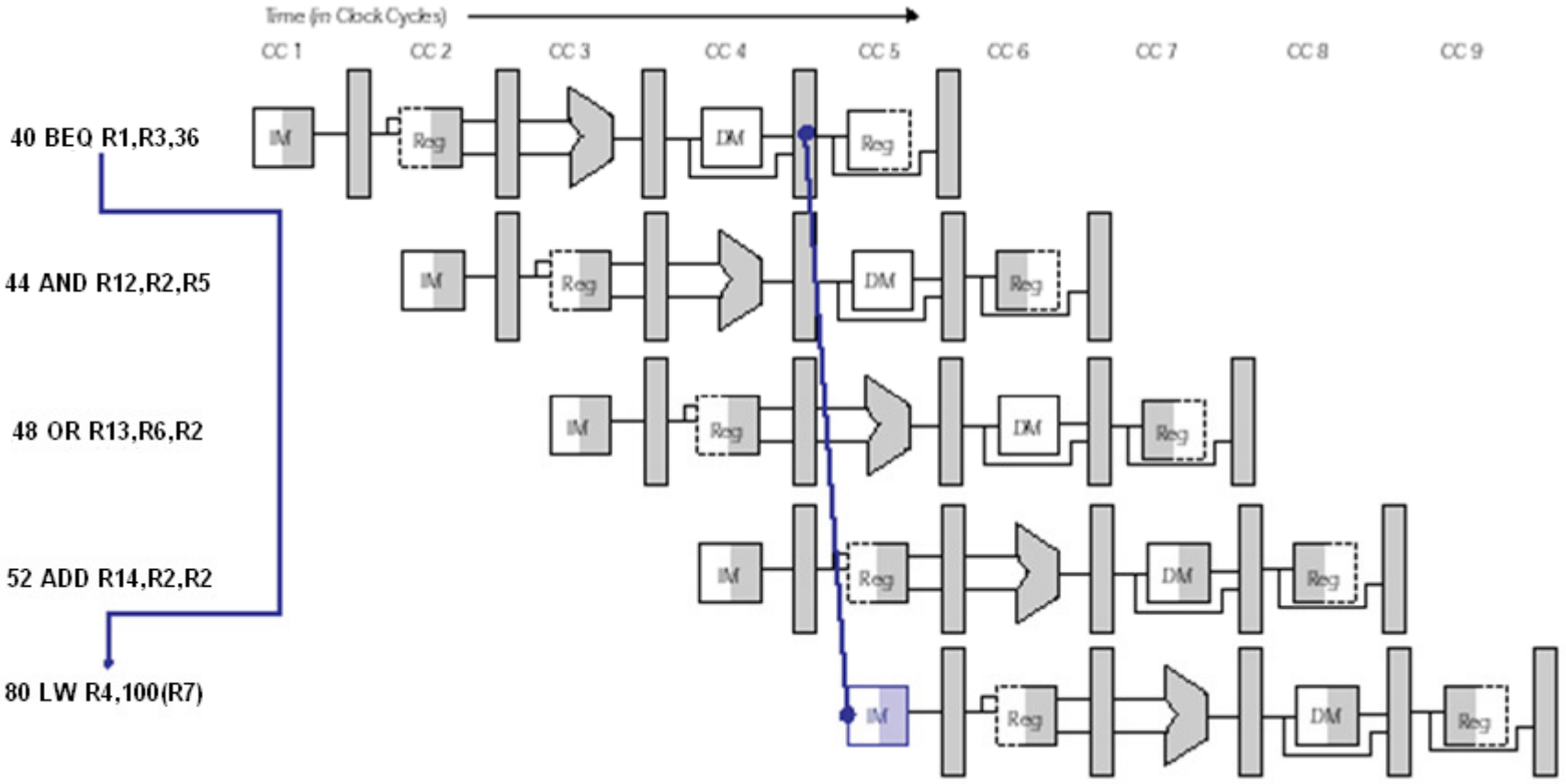
Perguntas: 1) Porque não saída de WB?

2) Porque buscar valor calculado em EX na Saída de DM?

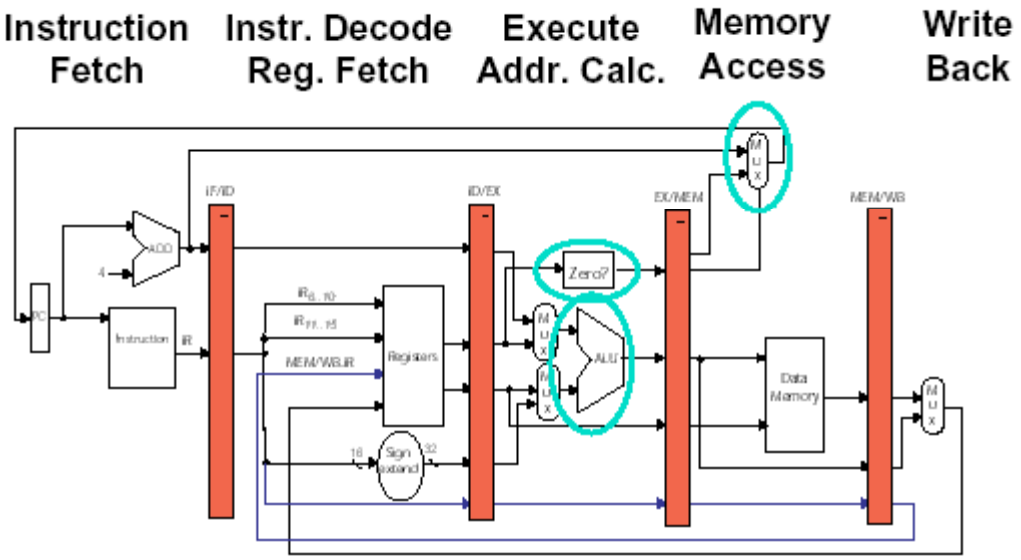
Resumo: Soluções para os Problemas no Pipeline

- Dependência (Conflito) de Recursos
  - Tabela de Recursos, Multiplicação mais HW
- Dependência de Dados
  - RAW: Read After Write
    - Auxílio dos compiladores
    - Encaminhamento de dados (bypassing)
  - WAW, WAR ?
- Dependência de Controle
  - ?

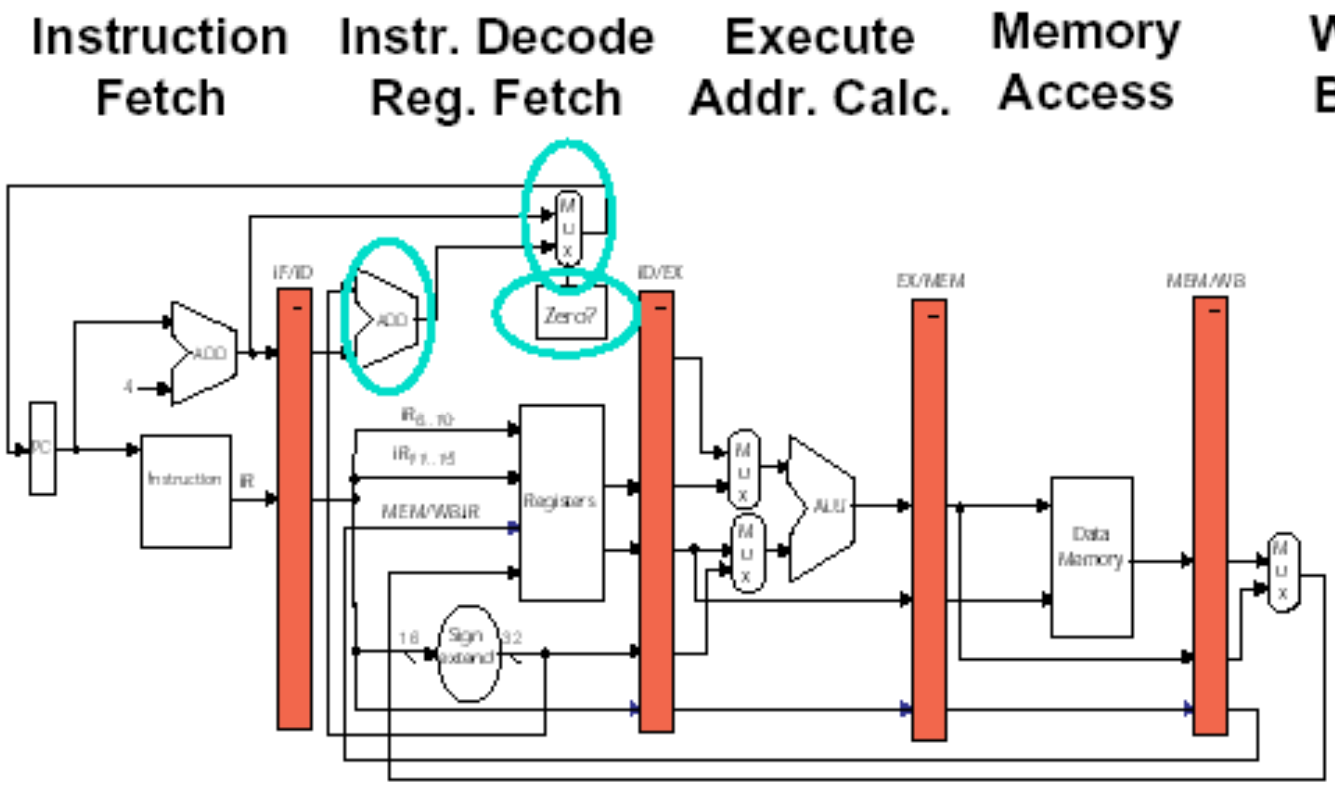
O atraso do pipeline em Desvios Condicionais - Como diminuir o atraso?



Atraso em Desvios Condicionais



Diminuindo para um ciclo de atraso



Seria possível eliminar o atraso em desvios ?

Dependência de Controle- Predição de desvios

- Consiste em **adivinhar** o resultado da condição de desvio e proceder como se a **adivinhação estivesse correta**.
  - **Estado da CPU não pode ser afetado se houver erro na predição**

- Predição de desvios é muito boa para o desempenho quando há boas taxas de acerto de predição; em muitos casos isto é possível:

- **Exemplos:**
  - Teste de fim de laço sempre dá falso, exceto no final;
  - Procuras falham em todas as iterações, exceto possivelmente na última.

- Geralmente utilizado em processadores superescalares, iremos detalhar tal técnica ao tratarmos deste assunto

Desvio Retardado

- Adota-se instruções de desvio que **permitem** executar uma ou mais **instruções subseqüentes** antes de desviar.
- **Compiladores com otimizadores** devem escolher as instruções a serem colocadas após essas instruções de desvio; elas devem ser efetivamente executadas.
- Considerando penalidade de um stall (bolha) por desvio, no diagrama abaixo a instrução I5 poderia ser executada depois de I6, **antes do resultado da condição**

WB					**I1**	**I2**	**I3**	**I4**	**I5**	**I6**		**I19**	**I20**	**I21**
M				**I1**	**I2**	**I3**	**I4**	**I5**	**I6**		**I19**	**I20**	**I21**	**...**
EX			**I1**	**I2**	**I3**	**I4**	**I5**	**I6**		**I19**	**I20**	**I21**	**...**	**...**
ID		**I1**	**I2**	**I3**	**I4**	**I5**	**I6**		**I19**	**I20**	**I21**	**...**	**...**	**...**
IF	**I1**	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I19**	**I20**	**I21**	**...**	**...**	**...**	**...**

Desvio Retardado

- Com I6 substituído por um desvio retardado (I6')...

WB					**I1**	**I2**	**I3**	**I4**	**I6'**	**I5**	**I19**	**I20**	**I21**	**...**
M				**I1**	**I2**	**I3**	**I4**	**I6'**	**I5**	**I19**	**I20**	**I21**	**...**	**...**
EX			**I1**	**I2**	**I3**	**I4**	**I6'**	**I5**	**I19**	**I20**	**I21**	**...**	**...**	**...**
ID		**I1**	**I2**	**I3**	**I4**	**I6'**	**I5**	**I19**	**I20**	**I21**	**...**	**...**	**...**	**...**
IF	**I1**	**I2**	**I3**	**I4**	**I6'**	**I5**	**I19**	**I20**	**I21**	**...**	**...**	**...**	**...**	**...**

Sumário

- Quais as influências de pipeline em:
  - IC ?
  - CPI ?
  - Clock ?
- Qual o limite para os melhoramentos em CPI?
  - Impossível ter CPI menor que 1 clock.

Sumário 2

- Pipeline
  - Traz um excelente ganho de desempenho em potencial
  - Traz grandes novos problemas (dependências estruturais, de controle e de dados)
  - Facilitado por instruções “bem comportadas”
    - Mesmo tamanho
    - Acesso a memória apenas em load/store
    - Poucos operandos
- Pipeline não é nada sem Controle!