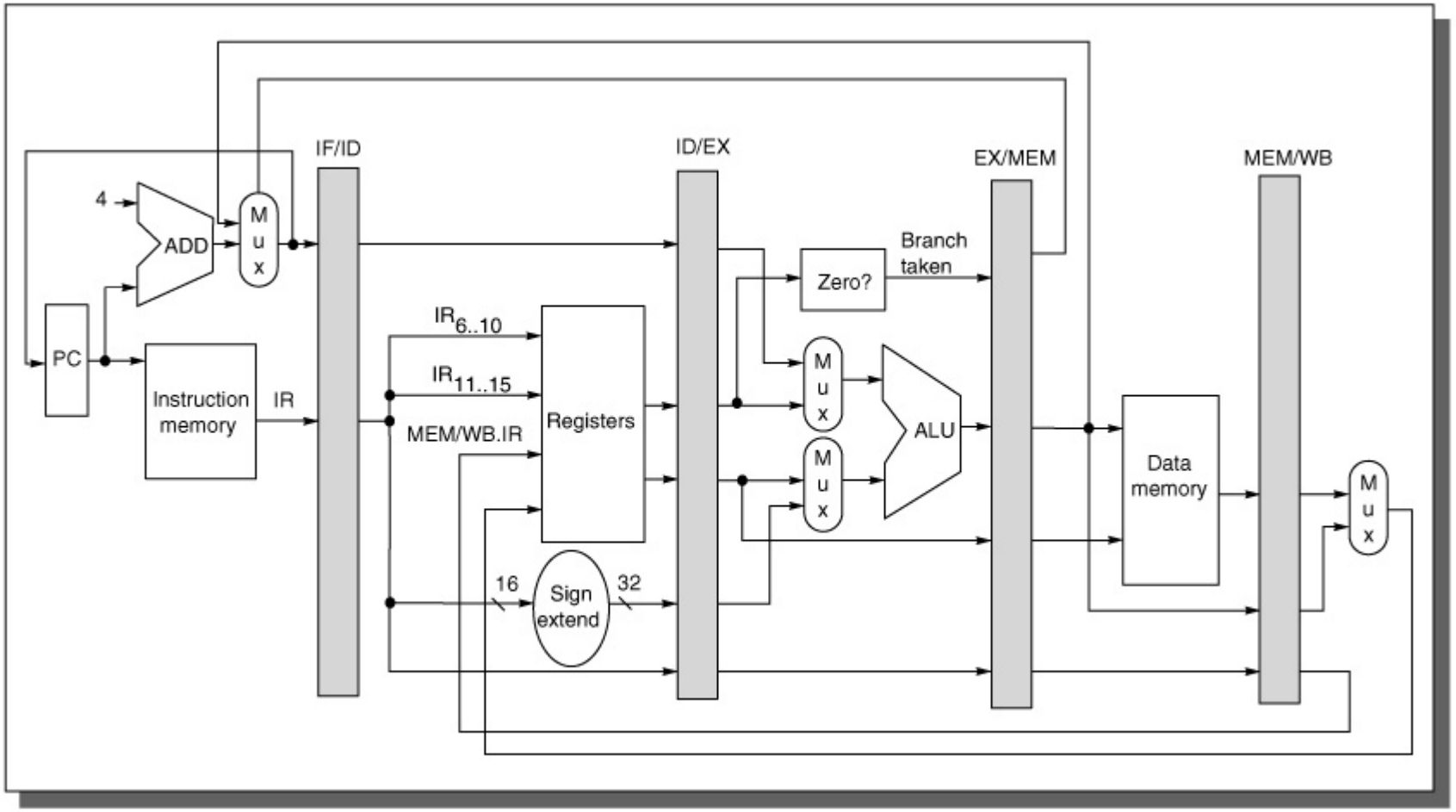
DCC007 – Organização de Computadores II

Aula 5 – Pipelining
Hazard de Controle

Prof. Omar Paranaiba Vilela Neto

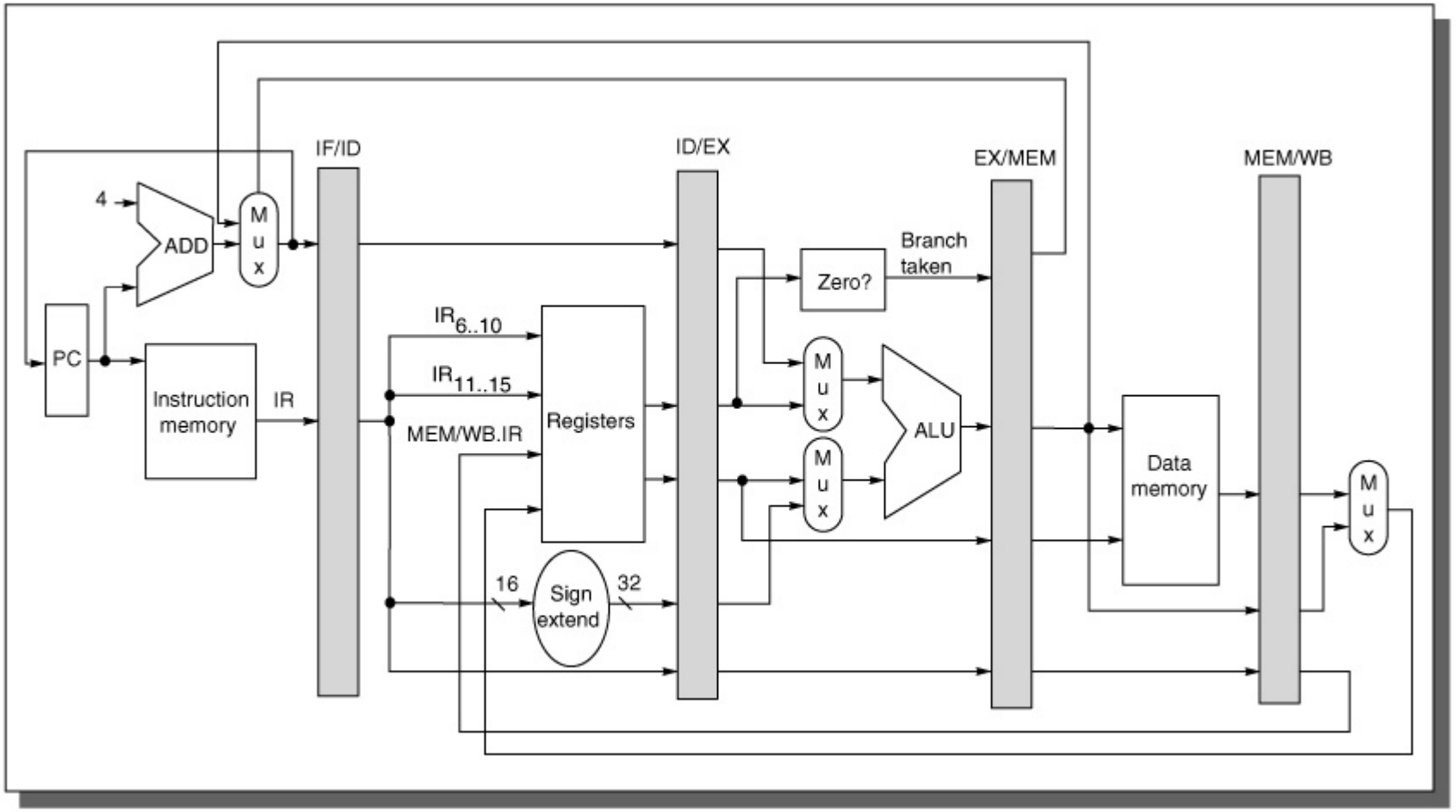
Datapath do MIPS com Pipeline



Representação Esquemática do *Pipeline*

Instrução	1	2	3	4	5	6	7	8	9
i	IF	ID	EX	MEM	WB				
i+1		IF	ID	EX	MEM	WB			
i+2			IF	ID	EX	MEM	WB		
i+3				IF	ID	EX	MEM	WB	
i+4					IF	ID	EX	MEM	WB

Hazards de Controle
Stall de 3 ciclos



Hazard de Controle
Stall de 3 ciclos

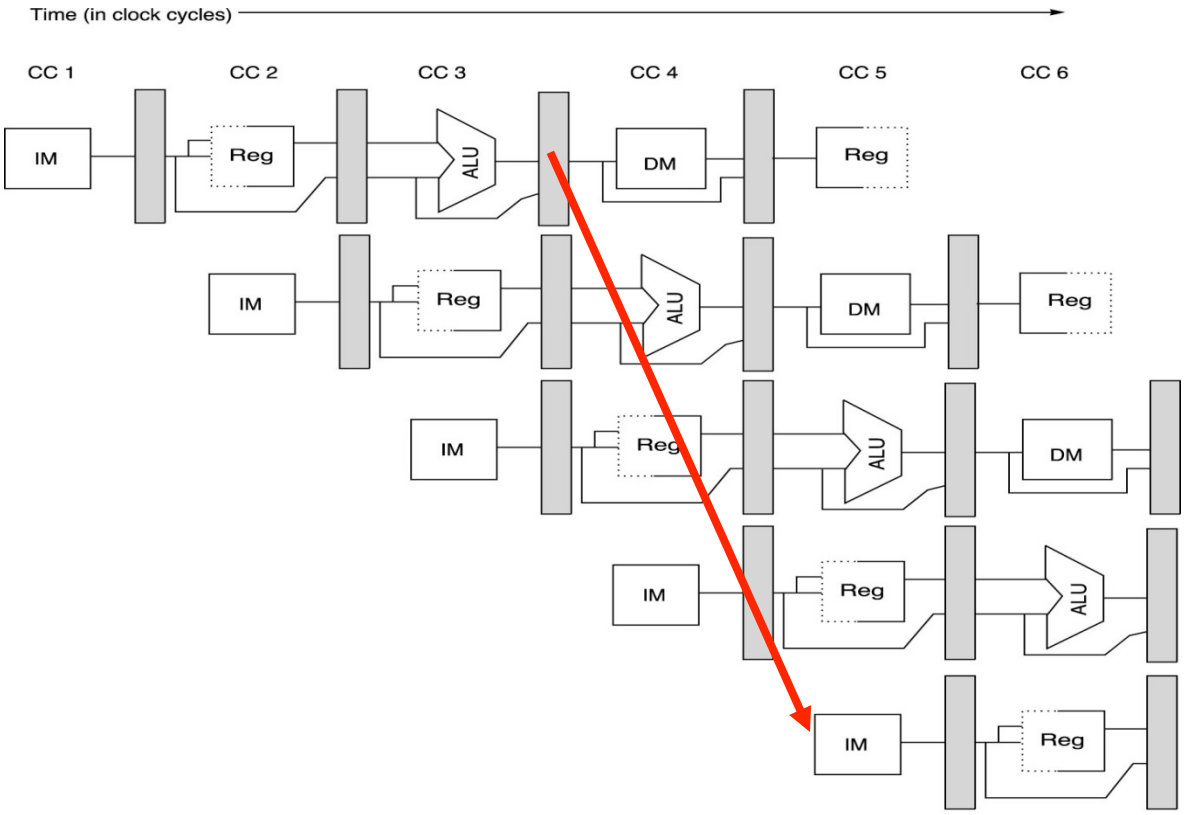
BEQ R1,L

DADD R2,R3,R4

DSUB R5,R6,R7

DADD R7,R2,R5

AND R2,R3,R4



Efeito de Branches
Solução “Inocente”

Instrução	1	2	3	4	5	6	7	8	9
branch	IF	ID	EX	MEM	WB				
i+1		IF	-	-	IF	ID	EX	MEM	WB
I+2						IF	ID	EX	MEM

Vamos parar o *pipeline* até endereço ser calculado:

- Simples de implementar

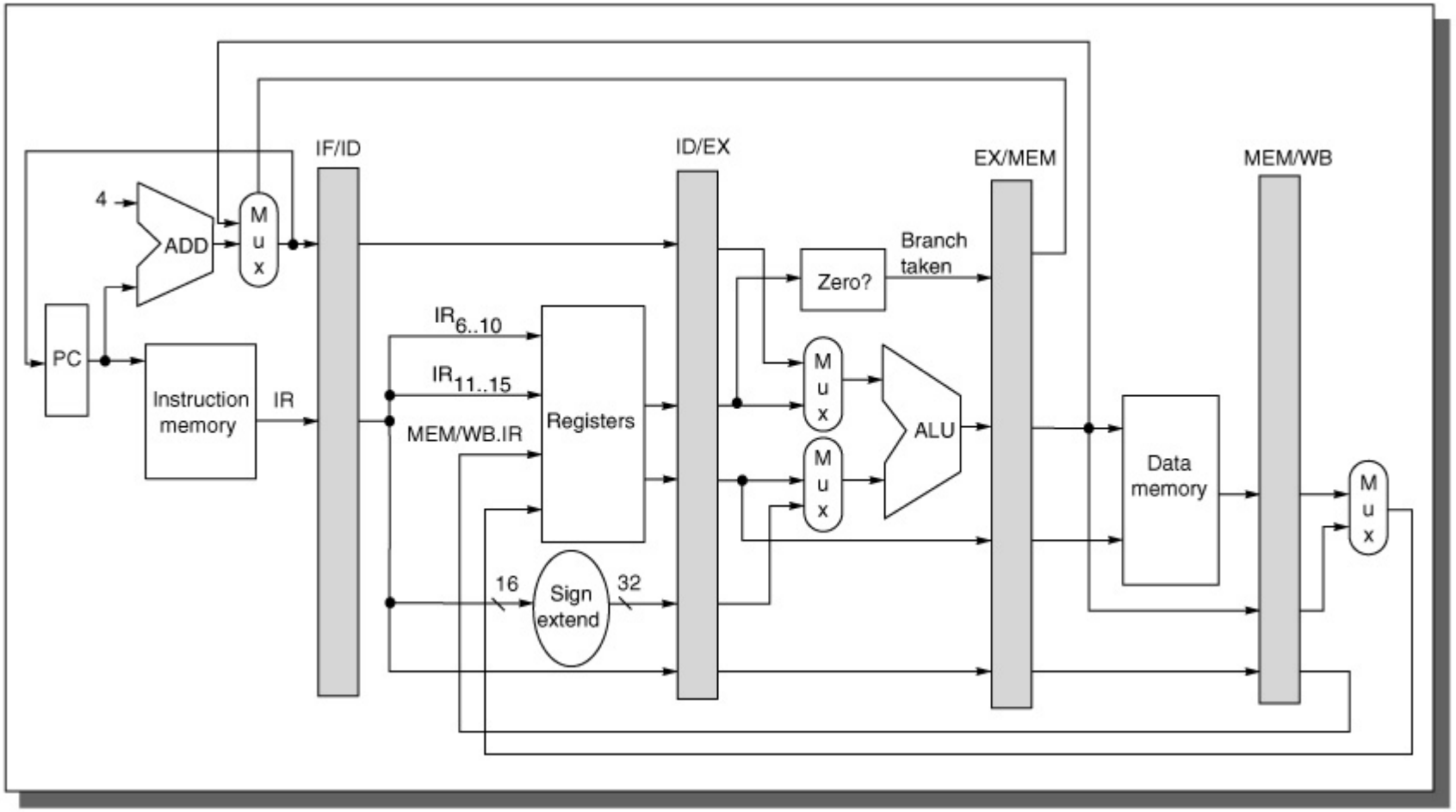
Impacto do Stall Devido a Branches

- Se CPI = 1, 30% de branches, Stall de 3 ciclos => novo CPI = 1.9!

1/2 do desempenho é perdido !!!

- **O que podemos fazer para melhorar?**

Hazards de Controle
Stall de 3 ciclos



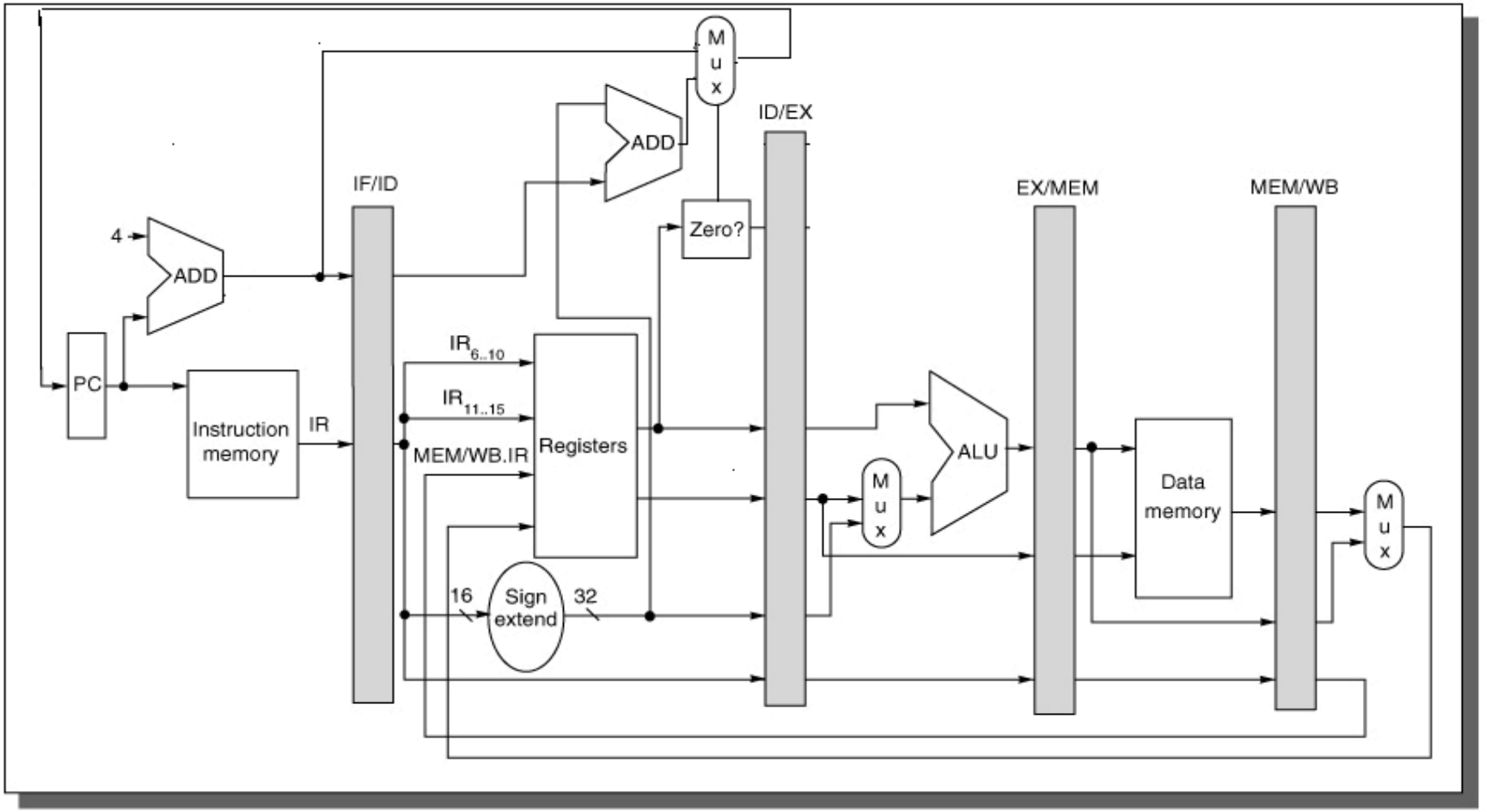
Impacto do Stall Devido a Branches

- Se CPI = 1, 30% de branches, Stall de 3 ciclos => novo CPI = 1.9!

  1/2 da performance é perdida !!!

- Solução em duas partes:
  - Determinar se branch é tomado ou não mais cedo, E
  - Calcular endereço de branch tomado mais cedo
- MIPS testa para branch se registrador é 0 ou não
- Solução:
  - Mover teste de 0 para estágio ID/RF
  - Aloque um somador para calcular novo PC no estágio ID/RF
  - **Reduz penalidade de 3 ciclos para 1!**

Datapath do MIPS com Pipeline



Características de Branches

- Programas inteiros
  - 13% das instrs. são branches condicionais para frente
  - 3% das instrs. são dos branches condicionais para trás
  - 4% das instrs. são branches incondicionais
- Programas FP´s
  - 7% das instrs. são branches condicionais para frente
  - 2% das instrs. são dos branches condicionais para trás
  - 1% das instrs. são branches incondicionais
- Dependências adicionais: precisamos saber quais branches são tomados ou não
  - 67% dos branches são tomados
  - 60% dos branches para frente são tomados (if-then-else)
  - 85% dos branches para trás são tomados (while)

Quatro Alternativas para Melhorar Desempenho com Branches

- #1: Para pipeline até sabermos se o branch será tomado ou não
- #2: Previsão de branch como não tomado
  - Continue executar instruções na sequência do pipeline
  - "Mate" execução das instruções que estão no pipe se branch for tomado
  - Vantagem se estado é alterado somente mais tarde (pode voltar atrás sem ter que desfazer nada)
  - +- 1/3 dos branches no MIPS não são tomados na média
  - PC+4 já é calculado para a próxima instrução
- #3: Previsão de branch como tomado
  - +- 2/3 dos branches são tomados na média
  - Mas no MIPS o endereço demora de qualquer jeito 1 ciclo para ser calculado (não há ganho)
    - Outras máquinas: destino conhecido antes

Previsão de Branches como não Tomados no MIPS

NT

Instrução	1	2	3	4	5	6	7	8	9
branch	IF	ID	EX	MEM	WB				
i+1		IF	ID	EX	MEM	WB			
I+2			IF	ID	EX	MEM	WB		

T

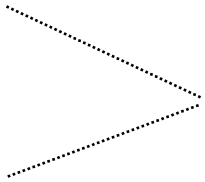
Instrução	1	2	3	4	5	6	7	8	9
branch	IF	ID	EX	MEM	WB				
t		IF	IF	ID	EX	MEM	WB		
t+1				IF	ID	EX	MEM	WB	

Quatro Alternativas para Melhorar Performance de Branches

#4: *Delayed Branch*

- Branch completa todas as instruções sucessoras

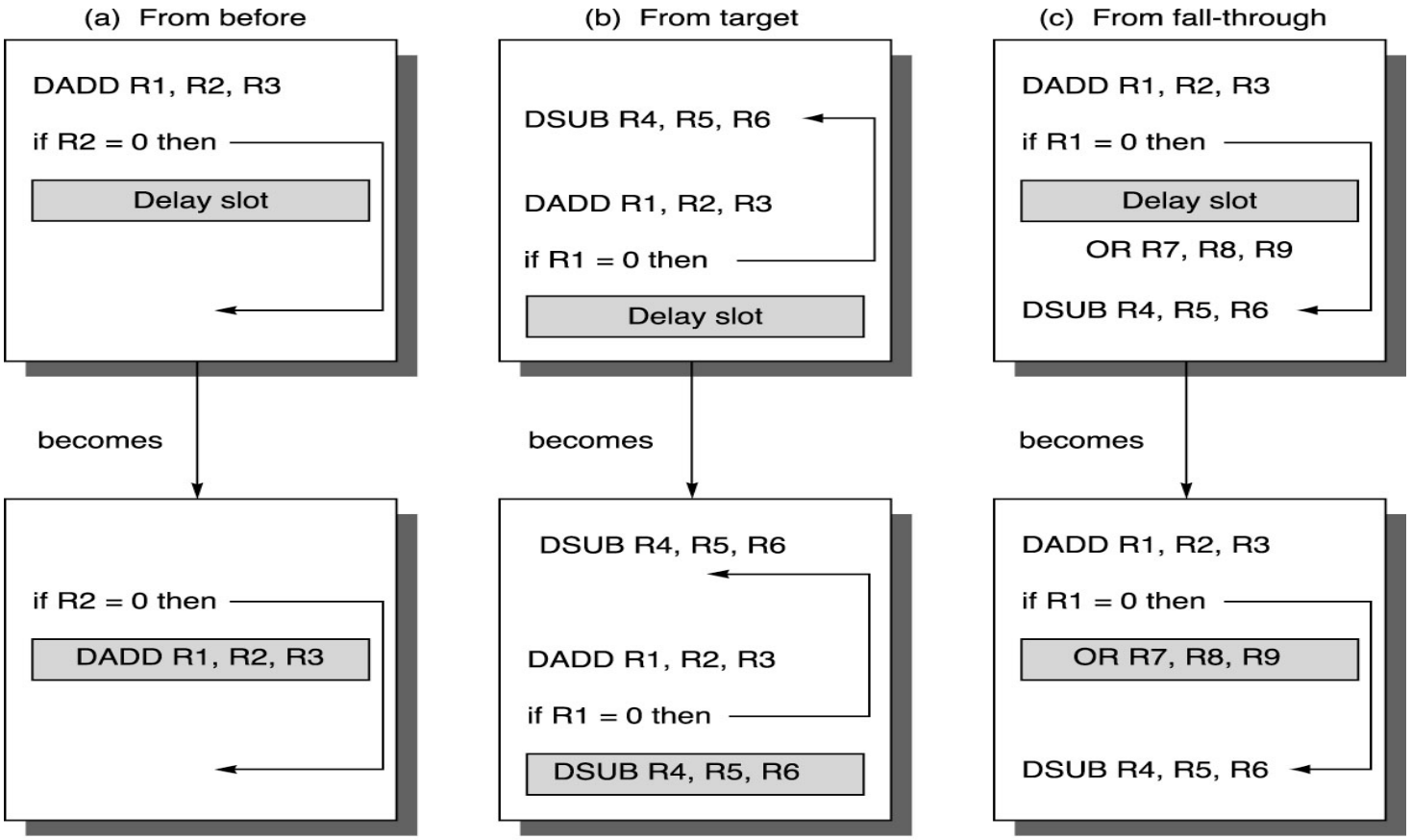
```
branch instruction
 sequential successor1
 sequential successor2
 ........
 sequential successorn
branch target if taken
```



Branch delay* de tamanho *n

- Delay de 1 slot permite decisão para o endereço destino em um pipeline de 5 estágios
- MIPS usa delay de 1 slot

Delayed Branch



Delayed Branch

- Efetividade do compilador em preencher branch delay slot único:
  - Preenche aproximadamente 60% dos branch delay slots
  - Cerca de 80% das instruções executadas realizam código útil no programa
  - Cerca de 50% (60% x 80%) dos slots são preenchidos eficazmente
- *Cancelling branches*: permite compilador preencher a instrução e decidir quando sequência será cancelada

Avaliando Alternativas de Branch do MIPS 4000

$$\text{Pipeline speedup} = \frac{\text{Pipeline depth}}{1 + \text{Branch frequency} \times \text{Branch penalty}}$$

Branch	***incondicional***	***ñ tomado***	***tomado***
Para pipe	2	3	3
Assume tomado	2	3	2
Assume não tomado	2	0	3

Branch	***incondicional***	***ñ tomado***	***tomado***	***soma***
frequência	4,00%	6,00%	10,00%	20,00%
Para pipeline	0,08	0,18	0,30	0,56
Assume tomado	0,08	0,18	0,20	0,46
Assume não tomado	0,08	0,00	0,30	0,38

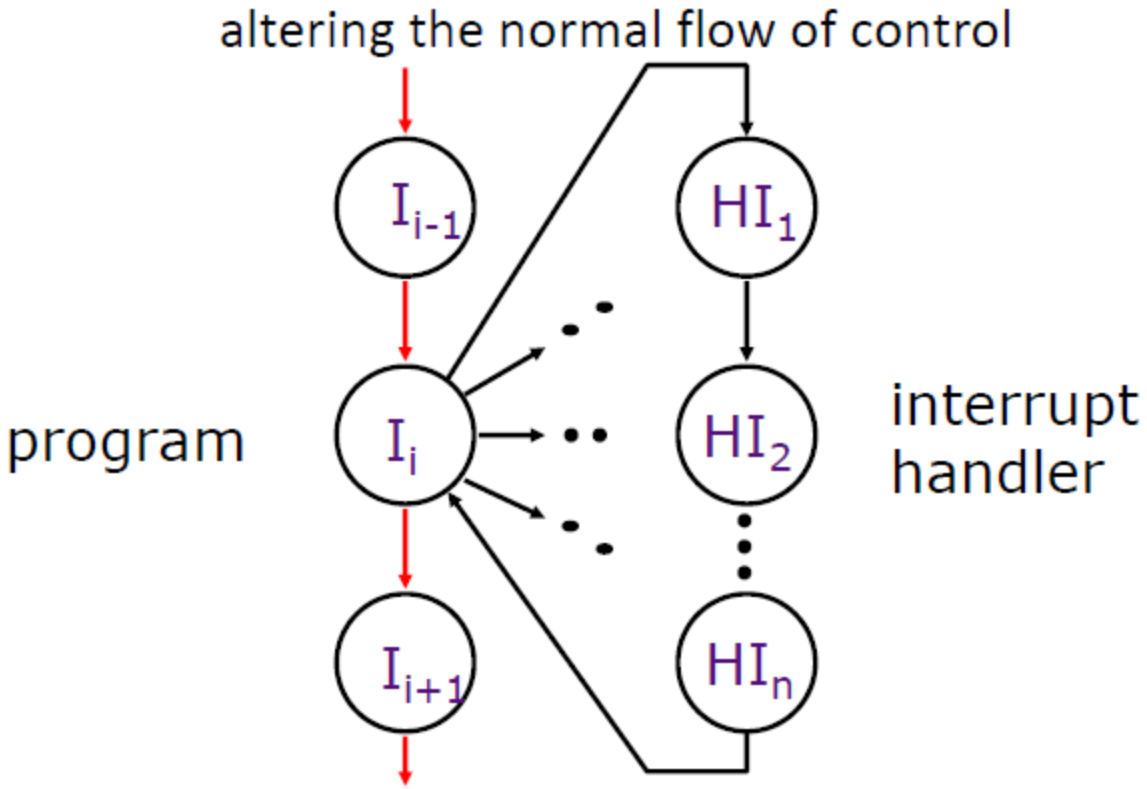
Complicações no Pipeline

- O que vimos até agora foi a parte fácil de pipelines
- O que torna pipeline difícil: interrupções
  - Não sabemos quando o estado da instrução pode ser salvo

Tipos de Exceção

- Operação de I/O
- Chamar rotina do S.O. a partir de programa de usuário
- Executando em modo de depuração (*single step*)
- Breakpoint
- Overflow aritmético
- Erro em aritmética de FP
- Page fault
- Acesso não alinhado
- Violação de proteção de memória
- Instrução não implementada
- Erro de hardware
- Falha na alimentação

Interrupção



Um **evento** interno ou externo que **necessita ser tratado por outro programa** (externo). O evento é **inesperado ou raro** do ponto de vista do programa.

Interrupção

- Assíncrona: um evento **externo**
  - Requisição de entrada-saída;
  - Tempo expirado;
  - Interrupção de energia;
  - Falha de hardware.
- Síncrona: um evento **interno**
  - Opcode indefinido, instrução com privilégio;
  - Overflow, exceção de FPU;
  - Acesso de memória desalinhado;
  - Exceção de memória virtual;
  - Exceção de software.

Tipos de Exceção

- Síncrona ou assíncrona
- Requisitado pelo usuário
- Mascarável ou não
- Dentro ou entre instruções
- *Resume* ou *terminate*

Parando e Re-executando Instruções

- Dentro da instrução
  - Força instrução de armadilha no pipeline
  - Até armadilha ser executada, desligue todas as escritas (salvamento de estado)
  - Rotina de SO primeiramente salva PC para reinicialização
- E se máquina tiver delayed branches?
  - Temos que salvar mais de um PC
  - Não poderemos recriar estado da máquina
- E se máquina tiver instruções que salvem o estado antes do final?
  - Vamos ver mais adiante

Complicações no Pipeline

- Exceções simultâneas em mais de um estágio do pipeline
  - Load com page fault para busca de dados em MEM
  - Add com page fault para busca de instrução em IF
  - Falha de Add acontecerá antes da falha do Load
- Solução #1
  - Vetor de status de interrupção por instrução
  - Atrase cheque até último estágio, então não deixe estado ser alterado para executar exceção
- Solução #2
  - Interrompa tão logo quanto exceção ocorra
  - Recomece tudo que está incompleto

Complicações no Pipeline

- Modos de endereçamento complexos e instruções complexas
- Modos de endereçamento: autoincremento causa mudança de estado durante execução
  - E se uma interrupção ocorrer? Precisa restaurar estado
  - Adiciona hazards de WAR e WAW
- Instruções de movimentação de blocos de memória
  - Precisa lidar com page faults múltiplas
  - Executam por muitos ciclos: salvam estado intermediário (8086)
- Condition Codes

Complicações no Pipeline

- Divide, Square Root levam 10X a 30X mais tempo que Add
  - O que acontece com interrupções?
  - Adiciona hazards de WAR e WAW já que instruções não ficam no pipeline pelo mesmo tempo

Interrupções Precisas

- Pipeline pode ser interrompido de tal forma que instruções anteriores a exceção completam e instruções posteriores à exceção podem ser reinicializadas
  - Isto é possível?
    - Não, senão não estaríamos vendo isso aqui
  - Ex: ADDF F3,F3,F4
    - Durante execução dessa instrução, F3 é escrito nos passos intermediários (FPs rodam por muitos ciclos)
  - Alpha, Power-2 e R8000 possuem dois modos de operação
    - c/ interrupções precisas: mais lento
    - s/ interrupções precisas: velocidade máxima do processador

Interrupções no MIPS

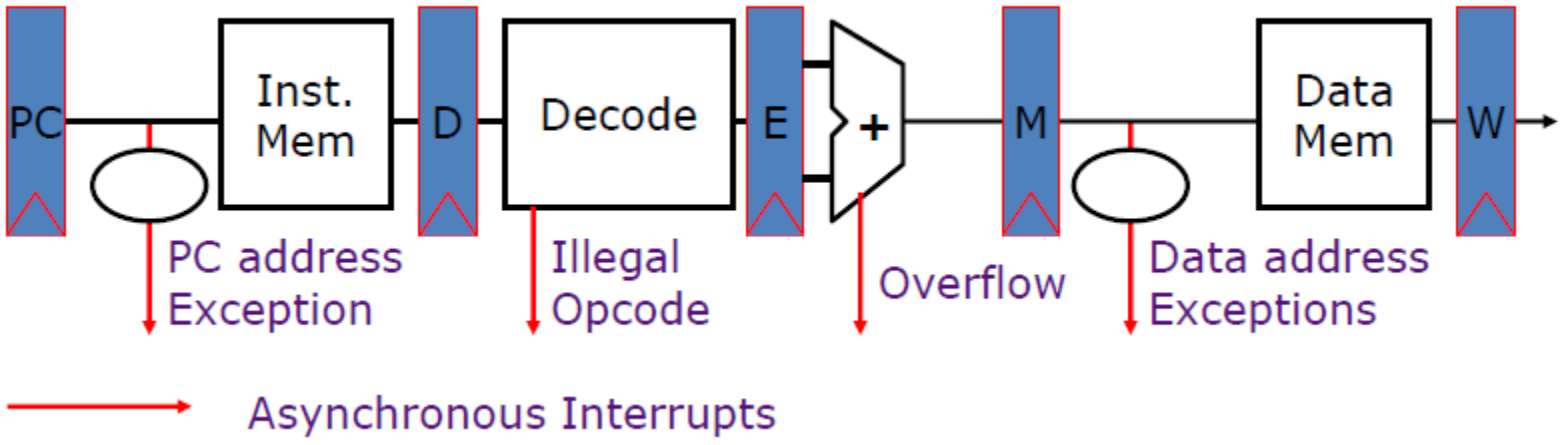
- Interrupções: 5 instruções executando nos 5 estágios do pipeline
  - Como parar o pipeline?
  - Como recomeçar o pipeline?
  - Quem causou a interrupção?

Interrupções no MIPS

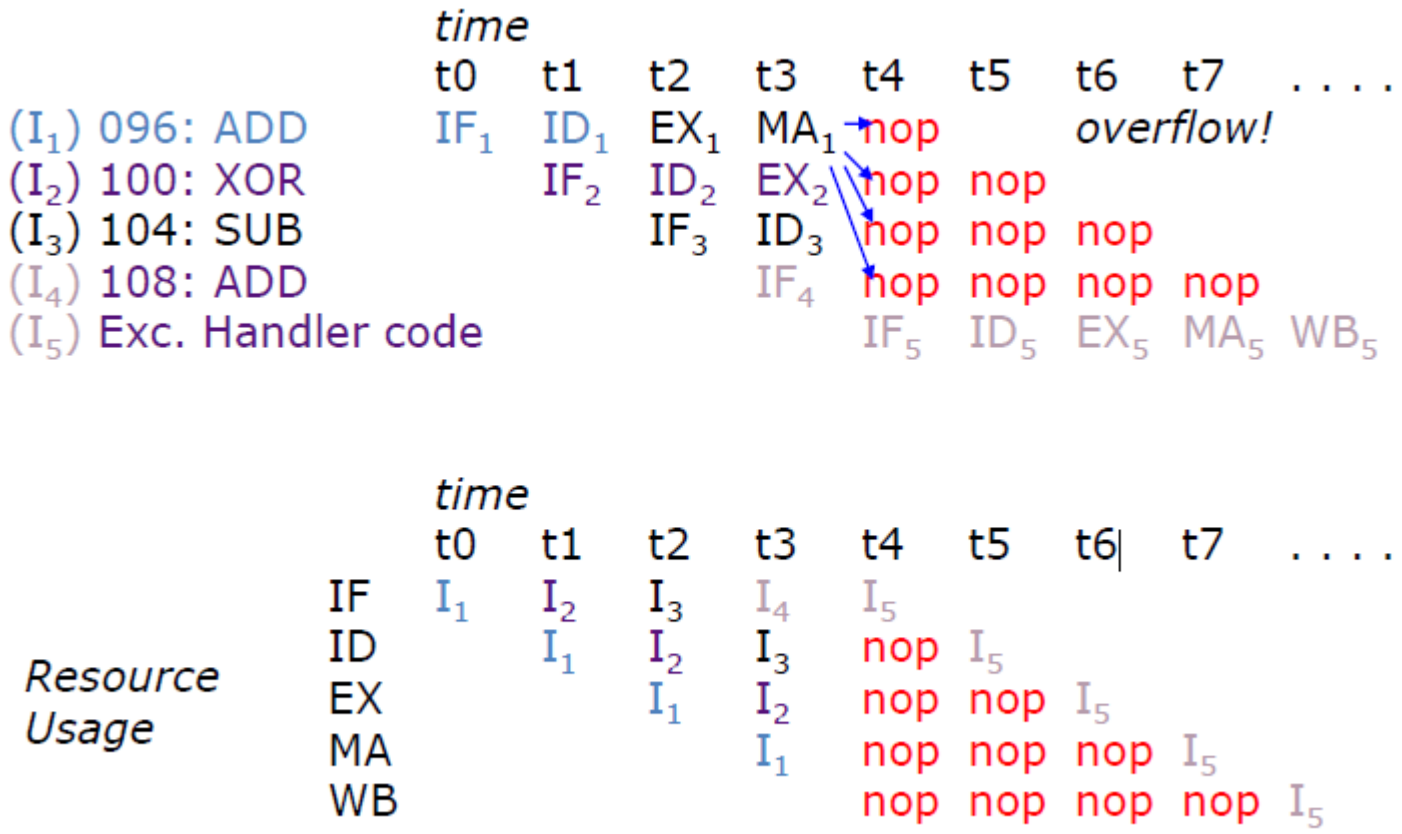
Estágios e Causas das Interrupções

- IF
  - Page fault na busca da instrução
  - Acesso não alinhado
  - Violação de proteção de memória
- ID
  - Código da instrução inválido e/ou protegido
- EX
  - Erro de operação da ALU
- MEM
  - Page fault na busca do dado
  - Acesso não alinhado
  - Violação de proteção de memória

Interrupções no MIPS



Interrupções no MIPS



Resumo de Introdução ao Pipeline

- Hazards limitam desempenho
  - Estrutural: falta de recursos de hardware
  - Dados: necessita de forwarding e escalonamento de instruções
  - Controle: avaliação do PC mais cedo, *delayed branch*, previsão do branch
- Aumento da profundidade causa maior impacto com hazards
- Interrupções, conjunto de instruções e FPs fazem pipeline mais difícil
- Compiladores reduzem penalidades de hazards de controle e dados