



INSTITUTO SUPERIOR TÉCNICO

MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE  
COMPUTADORES

ARQUITECTURAS AVANÇADAS DE COMPUTADORES

**Descrição do processador  $\mu$ Risc a funcionar em  
pipeline**

Guilherme Branco Teixeira	n.º 70214
Maria Margarida Dias dos Reis	n.º 73099
Nuno Miguel Rodrigues Machado	n.º 74236

Lisboa, 7 de Maio 2014

# Índice

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Métodos de resolução para dependências e conflitos</b>	<b>1</b>
2.1	Conflitos de dados, <i>data hazards</i> . . . . .	1
2.2	Conflitos de controlo, <i>control hazards</i> . . . . .	1
2.3	Conflitos de controlo, <i>control hazards</i> . . . . .	2
<b>3</b>	<b>Estrutura do Processador</b>	<b>2</b>
<b>4</b>	<b>Conclusões</b>	<b>2</b>

---

<sup>1</sup>É de referir que as imagens e tabelas não foram colocadas em anexo de modo a permitir uma melhor compreensão do relatório.

# 1 Introdução

Com este trabalho laboratorial pretende-se projectar um processador  $\mu$ Risc com funcionamento *pipelining*. O processador possui 4 andares de *pipelining*, no primeiro andar é feito o *instruction fetch* (IF), no segundo andar é feito o *instruction decode* (ID) e o *operand fetch* (OF), no terceiro andar são executadas operações da ALU (EX) e de acesso à memória de dados (MEM) e, por fim, no quarto é feita a escrita no banco de registos, o *write back* (WB). Com o funcionamento em *pipelining* poderá correr dois tipos de conflitos, dados (*data hazards*) e de controlo (*control hazards*).

## 2 Métodos de resolução para dependências e conflitos

Será apresentado em primeiro lugar, os métodos e técnicas de resolução dos conflitos de controlo e dados. E em segundo lugar quais as técnicas utilizadas.

### 2.1 Conflitos de dados, *data hazards*

Conflito que surge quando uma instrução depende dos resultados de uma instrução anterior, de forma a afectar o resultado obtido pela linha de processamento.

- **Solução 1:** Bloqueio dos andares do pipeline, *stall*, até que os dados correctos estejam disponíveis;
- **Solução 2:** Se o dado correcto existir algures no pipeline, estabelece-se um bypass para o andar correcto, aplica-se a técnica de *forwarding*;
- **Solução 3:** Escalonar/reordenar as instruções, se a ordenação for feita pelo compilador, tem-se um escalonamento estático, se for feita pelo *hardware*, escalonamento dinâmico;

Analisando as soluções apresentadas verificou-se que o escalonamento estático e dinâmico não seria a solução desejada devido a complexidade para um processador de 4 andares comparativamente às outras anteriores. Ponderou-se inicialmente a utilização de *stalls* devido à facilidade de implementação mas devido ao inconveniente de reduzir o número médio de instruções por ciclo, optou-se pela segunda solução de forma a aumentar o número médio de instruções por ciclo como também a complexidade de implementação.

### 2.2 Conflitos de controlo, *control hazards*

Conflito que surge quando uma instrução de controlo condicional depende dos resultados de uma instrução anterior, de uma forma a impedir uma predição correcta.

- **Solução 1:** BTB, foi contruída uma BTB com 9 bits de largura e apenas um bit de predição (BPB);
- **Solução 2:** *forwarding* de flags, após se obter o resultado necessário para a predição, estabelece-se um bypass para verificar a condição do salto;

### **3 Estrutura do Processador**

### **4 Conclusões**

## Todo list