

INSTITUTO SUPERIOR TÉCNICO

MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE
COMPUTADORES

ARQUITECTURAS AVANÇADAS DE COMPUTADORES

**Descrição do processador μ 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, 10 de Maio 2015

Índice

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

1 Introdução

Com este trabalho laboratorial pretende-se projectar um processador μ Risc com funcionamento em *pipeline*. 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* podem ocorrer dois tipos de conflitos - de dados (*data hazards*) e de controlo (*control hazards*).

2 Métodos de resolução para dependências e conflitos

Em primeiro lugar apresentam-se os métodos e técnicas de resolução dos conflitos de controlo e dados. Em segundo lugar quais as técnicas que foram de facto utilizadas.

2.1 Conflitos de dados: *data hazards*

Um conflito de dados surge quando uma instrução depende dos resultados da instrução anterior, de forma a afectar o resultado obtido pela linha de processamento.

referir que no
nosso caso so
ha conflitos do
tipo RAW

2.2 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, aplicando 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;

2.3 Conflitos de controlo: *control hazards*

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

nao impede a
prediccao

2.4 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, uma tabela que contém informação sobre os saltos de forma a prever se estes são *taken* ou *not-taken*, previsão esta que é realizada por uma BPB (composta por 1 ou 2 bits), diminuindo assim o número de ciclos desperdiçados em instruções do tipo controlo;
- **Solução 2:** *forwarding* de flags, após se obter o resultado necessário para a predição, estabelece-se um bypass para verificar a condição do salto;

explicar o que
é uma BTB,
explicar BPB

3 Estrutura do Processador

4 Conclusões

Todo list

	referir que no nosso caso so ha conflitos do tipo RAW	1
	nao impede a prediccao	1
	explicar o que é uma BTB, explicar BPB	2