Instituto Superior Técnico

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Arquitecturas Avançadas de Computadores

Descrição do processador μRisc 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

[1] 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 μRisc 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 uma forma que afecta o resultado obtido pela linha de processamento. De seguida está representado 3 soluções possíveis:

- **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 .

2.2 Conflitos de controlo, *control hazards*

3 Estrutura do Processador

4 Conclusões

Todo list