Instituto Superior Técnico

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Arquitecturas Avançadas de Computadores

Simulação de um processador μRisc

Maria Margarida Dias dos Reis	n.º 73099
Nuno Miguel Rodrigues Machado	n.º 74236

Lisboa, 29 de Março 2014

Índice

1 Introdução

Com este trabalho laboratorial pretende-se projectar um processador μRisc, de 16 *bits* com arquitectura RISC. O processador possui 8 registos de uso geral e 42 instruções. O projecto do processador é feito com recurso a uma linguagem de descrição de *hardware* - VHDL.

2 Características do Processador

O processador elaborado foi simulado para uma placa Artix 7 e tem as seguintes características:

- 16 *bits*;
- 8 registos de uso geral de 16 *bits* de largura (R0, ..., R7);
- 42 instruções;
- instruções de 3 operandos;
- organização de dados na memória do tipo *big endian*;
- uma memória ROM de 8 KBytes (4k endereços $\times$ 2 *bytes*) endereçada com palavras de 12 *bits* utilizada para as instruções/programa e uma memória RAM de 8 KBytes (4k endereços $\times$ 2 *bytes*) endereçada com palavras de 12 *bits* para os dados.

3 Estrutura do Processador

O processador μRisc que foi projectado encontra-se dividido em quatro andares - num 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 e último andar é feita a escrita no banco de registos, o *write back* (WB).

3.1 Primeiro Andar - IF

No primeiro andar a próxima instrução a ser executada é carregada da memória ROM.

3.2 Segundo Andar - ID e OF

3.3 Terceiro Andar - EX e MEM

3.4 Quarto Andar - WB

No último andar os diversos resultados possíveis são escritos no banco de registos - pode ser o resultado de uma operação da ALU, o resultado de uma operação sobre a memória (*load*), o carregamento de uma constante ou guardar em R7 o valor do próximo *program counter*. Como se pode ver na figura seguinte, a seleção de qual os resultados deve ser escrito é feita com recurso a um MUX 4:1.

explicar sinal de selecçao do mux

FIGURA

Uma vez seleccionado o resultado a escrever é preciso escolher qual o registo onde se pretende escrever esse mesmo resultado (registo WC).

Para instruções da ALU a escolha do registo onde se quer escrever o resultado final é feita com recurso aos *bits* 11 a 13 da instrução, assim como para operações de carregamento de constantes. Porém, para operações de transferência de controlo esses mesmos *bits* representam a operação a realizar, pelo que não basta utilizar os 3 *bits* referidos anteriormente para controlar um *decoder* que colocasse a *high* um dos 7 *enables* (que estão armazenados nos 7 *bits* de um vector), tal como pensado originalmente e como pode ser visto na figura abaixo.

FIGURA

Assim, para resolver o problema dos casos em que se pretende fazer um *jump* ou o *store* de dados na RAM é necessário criar um sinal que faça *override* ao *enable* que o *decoder* colocou a *high*, permitindo o sinal de *override* colocar o *enable* a *low*, tal como pretendido, para que não se escreva em nenhum registo.

exemplo de um jump que depois ia escrever num registo e nao era suposto. dizer que para stores tambem nao se quer escrever em nenhum reg

Todo list