**universidade federal de minas gerais**

**IMPLEMENTAÇÃO DE PROCESSADORES SUPERSCALARES**

**I2OI & IO2I**

Trabalho final da disciplina de ORGANIZAÇÃO DE COMPUTADORES II

**Alunos:**

Lecio Charlles

Breno Pimenta

Tiago Coelho Magalhães

|  |
| --- |
| BELO HORIZONTE  2020 |

Prof.Omar Paranaiba Vilela Neto

1. Introdução

Este trabalho prático consiste em descrever as partes fundamentais dos processadores superescalares utilizando a linguagem de descrição de hardware (*Hardware Description Language*, HDL) Verilog, que permite o modelamento de sistemas eletrônicos ao nível de circuito.

É sabido que processadores superescalares tem como objetivo maximizar a capacidade de execução de instruções por ciclo de clock (**CPI < 1**), o que se dá por meio de um paralelismo a nível de instruções. Os processadores superescalares podem ser caracterizados por realizar determinadas etapas do ciclo de processamento em ordem (*in-order*) ou fora de ordem (*out-of-order*). A Tabela 1 ilustra alguma destas possibilidades e evidencia a existência de blocos específicos desta tecnologia. É possível observar que a busca da instrução (*fetch*) ocorre sempre em ordem, pois, do contrário, teríamos uma aleatoriedade na execução do programa, o qual possui uma lógica programada sequencialmente.

|  |
| --- |
|  |

Tabela 1 - Tipos de processadores superescalares

É válido ressaltar que o desempenho dos processadores também depende do compilador utilizado, o qual é responsável por agendar as instruções de forma a minimizar as dependências e, consequentemente, evitar *hazards*.

O presente trabalho tem como proposta abordar a implementação dos processadores I2OI e IO2I. Um subconjunto de instruções do MIPS fora proposto, com alteração de *opcodes*, para a realização das operações lógicas, aritméticas, desvios e de memória (*load* e *store*), conforme Tabela 2.

|  |
| --- |
|  |

Tabela 2 - Definição de Instruções

1. Processadores Superescalares

Na sequência, serão apresentadas as arquiteturas projetadas para os processadores I2OI e IO2I e as decisões de projeto adotadas durante o planejamento.

Nessas duas arquiteturas, bem como em outras arquiteturas superescalares, várias instruções podem ser executadas simultaneamente e de forma independente umas das outras (desde que estejam em estágios diferentes do pipeline), dada a existência de um caminho para as instruções associadas a diferentes operações, como aritméticas, lógicas e desvios (seguindo pelo caminho de X0), de *load* (pelo caminho de L0 e L1), *store* (pelo caminho de S0), e de multiplicação (pelo caminho de Y0, Y1, Y2 e Y3). Como já comentado anteriormente, o disparo da execução ou a escrita do resultado na memória podem acontecer em ordem ou fora de ordem. Basicamente, são estas as características que diferenciarão os processadores que serão desenvolvidos neste trabalho.

* 1. Processador I2OI

A Figura 1 apresenta a arquitetura básica do processador superescalar I2OI.

|  |
| --- |
|  |

Figura 1 - Processador I2OI (teórico)

Além dos caminhos associados a execução das instruções propriamente ditas, a Figura 1 evidencia um conjunto de estágios iniciais, que inclui FETCH, responsável pela busca de instruções; um estágio de DECODE, que interpreta os diversos operandos associados às instruções programadas; bem como o estágio de ISSUE, o qual verifica os conflitos de recursos para determinar quais instruções podem ser disparadas em cada etapa. Para esse processador, as instruções serão disparadas em ordem, entretanto, um estágio posterior à execução das instruções, o WRITEBACK ocorre fora de ordem. Um último estágio de COMMIT, que atualiza a memória, acontece em ordem.

Um ponto a ser destacado é a existência de um bloco SB, denominado *scoreboard*, associado ao módulo de ISSUE. Para o processador I2OI, e levando em consideração o compilador que organiza as instruções de forma a evitar dependências, como descrito anteriormente, o scoreboard construído para o projeto objetiva tratar *hazards* estruturais, construindo uma tabela de controle do uso e disponibilidade dos dados, ao longo do pipeline de execução.

A implementação do processador buscou evidenciar, de maneira visível, os mesmos blocos da Figura 1, e resultou na implementação vista na Figura 15 (Ap. A).

* + 1. Detalhamento dos Blocos

O projeto buscou apresentar os módulos principais da arquitetura da Figura 1, os quais serão detalhados a seguir, incluindo todos os submódulos existentes.

* + - 1. FECTH

O módulo é apresentado na Figura 2. Sua funcionalidade é a busca de instruções e, por isso, associado a ele está o submódulo contador de programa (*Program Counter*, PC), que contém o endereço da instrução que está sendo executada no momento. Conforme cada instrução é buscada, o contador do programa aumenta seu valor armazenado em 4. Existe, entretanto, uma lógica associada que decide se o endereço do PC será atualizado com o valor do “fluxo normal” ou do endereço de um eventual desvio.

|  |
| --- |
|  |

Figura 2 - Módulo FETCH

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Input/Output | Tamanho | Função |
| clk | Input | 1 | Sinal de clock global |
| rst | Input | 1 | Sinal de reset global |
| stall | Input | 1 | Sinal de paralisação de pipeline (atraso na execução de uma instrução para resolver um *hazard*) |
| pc\_src | Input | 1 | Informar se o PC será atualizado com o valor de desvio ou com +4 |
| add\_res | input | 32 | Endereço em caso de desvio |
| d\_inst | output | 32 | Instrução buscada |
| d\_pc | output | 32 | Valor de endereço da instrução atual |

Tabela 3 - Descrição de Sinais do Módulo FETCH

* + - 1. DECODE

O módulo decodificador de instruções é apresentado na Figura 2.

|  |
| --- |
|  |

Figura 3 - Módulo DECODE

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Input/Output** | **Tamanho** | **Função** |
| clk | Input | 1 | Sinal de clock global |
| rst | Input | 1 | Sinal de reset global |
| stall | Input | 1 | Sinal de paralisação de pipeline (atraso na execução de uma instrução para resolver um *hazard*) |
| permitEscrita | Input | 1 | Sinal de controle para escrita no banco de registradores |
| inst | input | 32 | Instrução a ser decodificada |
| pc | Input | 32 | Endereço no contador de programa |
| writedata | Input | 32 | Valor a ser gravado no banco de registradores |
| w\_regC | Input | 5 | endereço para escrita nos registradores, regC de inst tipo R e destino do load |
| e\_rd1 | output | 32 | Dado associado ao primeiro operando |
| e\_rd2 | output | 32 | Dado associado ao segundo operando |
| e\_sigext | output | 32 | Valores imediatos/externos |
| e\_pc | output | 32 | Endereço no contador de programa |
| e\_inst1 | output | 5 | Endereço primeiro operando |
| e\_inst2 | output | 5 | Endereço segundo operando |
| e\_inst3 | output | 5 | Endereço de destino, caso seja tipo R |
| e\_aluop | output | 2 | Operador da ALU |
| e\_alusrc | output | 1 | Sinal de controle |
| e\_regdst | output | 1 | Sinal de controle |
| e\_regwrite | output | 1 | Sinal de controle |
| e\_memread | output | 1 | Sinal de controle |
| e\_memtoreg | output | 1 | Sinal de controle |
| e\_memwrite | output | 1 | Sinal de controle |
| e\_branch | output | 1 | Sinal de controle |

Tabela 4 - Descrição de Sinais do Módulo DECODE

Os submodulos do DECODE são mostrados na Figura 4, sendo os principais, *RegisterBank*, que corresponde a um banco de registradores; *Control*, responsável por disparar os sinais de controle para o processador; e *IDI*, um bloco intermediário que apenas replica os sinais de entrada para a saída, associados aos sinais de controle e do banco de registradores, que serão enviados ao próximo bloco: ISSUE.

|  |
| --- |
|  |

Figura 4 - Submodulos do Módulo DECODE

Os sinais de controle do módulo CONTROL são definidos de acordo com o *opcode* definido à princípio. Parte do código que define o estado dos sinais de controle para a instruções tipo R é mostrado Figura 5

|  |
| --- |
|  |

Figura 5 - Parte do código do módulo de controle

* + - 1. ISSUE & SCOREBOARD

O módulo ISSUE é apresentado na Figura 6. Sua funcionalidade é disparar as instruções para que possam ser executadas. Com relação ao processador I2OI, o disparo da execução das instruções acontece fora de ordem (*out-of-order*). Para tanto, o módulo conta com o módulo SCOREBOARD para agendar dinamicamente a execução das instruções quando não houver *hazards* e o hardware estiver disponível.

Uma decisão de projeto adotada para a implementação do disparo da instrução a ser executada é a criação de um sinal “**e\_select\_execute**” responsável por informar o caminho responsável pelo processamento da instrução, ou seja, através de X0 ou L0/L1 ou M0/M1/M2/M3 ou S0.

|  |
| --- |
|  |

Figura 6 - Módulo ISSUE

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Input/Output** | **Tamanho** | **Função** |
| clk | Input | 1 | Sinal de clock global |
| rst | Input | 1 | Sinal de reset global |
| i\_rd1 | Input | 32 | Dado associado ao primeiro operando |
| i\_rd2 | Input | 32 | Dado associado ao segundo operando |
| i\_sigext | Input | 32 | Valores imediatos/externos |
| i\_pc | Input | 32 | Endereço no contador de programa |
| i\_inst1 | Input | 5 | Endereço primeiro operando |
| i\_inst2 | Input | 5 | Endereço segundo operando |
| i\_inst3 | Input | 5 | Endereço de destino, caso seja tipo R |
| i\_aluop | Input | 2 | Operador da ALU |
| i\_alusrc | Input | 1 | Sinal de controle |
| i\_regdst | Input | 1 | Sinal de controle |
| i\_regwrite | Input | 1 | Sinal de controle |
| i\_memread | Input | 1 | Sinal de controle |
| i\_memtoreg | Input | 1 | Sinal de controle |
| i\_memwrite | Input | 1 | Sinal de controle |
| i\_branch | Input | 1 | Sinal de controle |
| e\_rd1 | output | 32 | Dado associado ao primeiro operando |
| e\_rd2 | output | 32 | Dado associado ao segundo operando |
| e\_sigext | output | 32 | Valores imediatos/externos |
| e\_pc | output | 32 | Endereço no contador de programa |
| e\_inst1 | output | 5 | Endereço primeiro operando |
| e\_inst2 | output | 5 | Endereço segundo operando |
| e\_inst3 | output | 5 | Endereço de destino, caso seja tipo R |
| e\_aluop | output | 2 | Operador da ALU |
| e\_alusrc | output | 1 | Sinal de controle |
| e\_regdst | output | 1 | Sinal de controle |
| e\_regwrite | output | 1 | Sinal de controle |
| e\_memread | output | 1 | Sinal de controle |
| e\_memtoreg | output | 1 | Sinal de controle |
| e\_memwrite | output | 1 | Sinal de controle |
| e\_branch | output | 1 | Sinal de controle |
| e\_select\_execute | output | 3 | informa o caminho responsável pela execução da instrução |
| stall | output | 1 | Sinal de stall |
| e\_num\_inst\_PRF | output | 5 | sinal associado a escrita in-order associado ao Reorder Buffer |

Tabela 5 - Descrição dos sinais do módulo ISSUE

Comojá mencionado, um submódulo denominado selectExPip é responsável por informar o caminho responsável pela execução da instrução, ou seja, através de X0 ou L0/L1 ou M0/M1/M2/M3 ou S0. Esse sinal, ainda, está associado ao quadro de scoreboard proposto, o qual indica o estado de cada unidade funcional. Para que seja possível indicar qual unidade de função escreverá os resultados da operação, um sinal “num\_inst\_PRF” direciona as escritas em ordem.

|  |
| --- |
|  |

Figura 7 - Submódulo SCOREBOARD e SelectExecute

* + - 1. EXECUTE

O conjunto de módulos associados a execução das instruções é apresentado na Figura 8 e os módulos específicos de X0 (verde), S0 (roxo), L0/L1 (preto), e M0/M1/M2/M3 (azul) estão destacados.

|  |
| --- |
|  |

Figura 8 - Módulo EXECUTE e seus respectivos caminhos de execução

Os sinais destes módulos são apresentados na Error: Reference source not found.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Input/Output** | **Tamanho** | **Função** |
| clk | Input | 1 | Sinal de clock global |
| rst | Input | 1 | Sinal de reset global |
| e\_in1 | Input | 32 | Dado associado ao primeiro operando |
| e\_in2 | Input | 32 | Dado associado ao segundo operando |
| e\_sigext | Input | 32 | Valores imediatos/externos |
| e\_pc | Input | 32 | Endereço no contador de programa |
| i\_inst1 | Input | 5 | Endereço primeiro operando |
| e\_inst2 | Input | 5 | Endereço segundo operando |
| e\_rc | Input | 5 | Endereço de destino, caso seja tipo R |
| i\_aluop | Input | 2 | Operador da ALU |
| e\_alusrc | Input | 1 | Sinal de controle |
| e\_regdst | Input | 1 | Sinal de controle |
| e\_select\_execute | Input | 3 | informa o caminho pela execução da instrução |
| x\_aluout | output | 32 | Resultado da operação |
| w\_novoPC | output | 32 | Novo valor do PC em caso de desvio |
| x\_register\_store\_adress | output | 5 | Endereço de destino para salvar resultado |
| w\_branchTomado | output | 1 | Sinal de controle |
| w\_regwrite | output | 1 | Sinal de controle |
| w\_pipe\_ativo | output | 1 | Endereço segundo operando |
| S0\_store\_value | output | 32 | Valor a ser armazenado |
| S0\_register\_store\_adress | output | 5 | Endereço de memória para salvar resultado |
| S0\_pipe\_ativo | output | 1 | Sinal de controle |
| Lx\_load\_aluout | output | 32 | Valor a ser carregado |
| Lx\_register\_store\_adress | output | 5 | Endereço para load |
| lx\_pipe\_ativo | output | 1 | Sinal de controle |
| regC\_out | output | 32 | Resultado da operação |
| endRc\_out | output | 5 | Endereço de memória para salvar resultado |
| mulx\_pipe\_ativo | output | 1 | Sinal de controle |

Tabela 6 - Sinais associados à etapa de execução

* + - * 1. X0

O módulo X0 é apresentado na Figura 9. É responsável pela execução das operações aritméticas, lógicas e desvios condicionais.

|  |
| --- |
|  |

Figura 9 - Módulo X0

* + - * 1. S0

O módulo S0 é apresentado na Figura 10, responsável pela execução das instruções de armazenamento (*store*).

|  |
| --- |
|  |

Figura 10 - Módulo S0

* + - * 1. L0 e L1

A seguir, os módulos para a execução das instruções de LOAD (Figura 11).

|  |
| --- |
|  |

Figura 11 - Módulo LOAD

* + - * 1. MULTIPLICAÇÃO

O caminho responsável pela execução das instruções de multiplicação é apresentado na Figura 12.

|  |
| --- |
|  |

Figura 12 – Módulos da execução da multiplicação

* + - 1. WriteBack

O módulo WriteBack é apresentado na Figura 13.

|  |
| --- |
|  |

Figura 13 - Módulo WriteBack

Os respectivos sinais são descritos na Error: Reference source not found.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Input/Output** | **Tamanho** | **Função** |
| clk | Input | 1 | Sinal de clock global |
| rst | Input | 1 | Sinal de reset global |
| fonte | Input | 32 |  |
| x\_aluout | Input | 32 | Resultado da operação de X0 |
| x\_register\_store\_adress | Input | 5 | Endereço de destino para salvar resultado deX0 |
| x\_pipe\_ativo | Input | 1 | Sinal de controle |
| Lx\_load\_aluout | Input | 32 | Valor a ser carregado |
| Lx\_register\_store\_adress | Input | 5 | Endereço para load |
| lx\_pipe\_ativo | Input | 1 | Sinal de controle |
| S0\_store\_value | Input | 32 | Valor a ser armazenado |
| S0\_register\_store\_adress | Input | 5 | Endereço de memória para salvar resultado |
| S0\_pipe\_ativo | Input | 1 | Sinal de controle |
| M\_regC | Input | 32 | Resultado da operação de Multiplicação |
| M\_endRegC | Input | 5 | Endereço de memória para salvar resultado de M |
| M\_pipe\_ativo | Input | 1 | Sinal de controle |
| dado\_regC | output | 32 | Valor a ser escrito na memória -> Commit |
| end\_RegC | output | 5 | Endereço a ser escrito na memória -> Commit |
| permiteEscrita | output | 1 | Sinal de controle |
| eh\_store | output | 1 | Indica escrita na memoria de dados, caso seja store |

Tabela 7 - Sinais associados ao módulo de WriteBack

* 1. Processador IO2I

A Figura 14 apresenta a arquitetura básica do processador superescalar IO2I.

|  |
| --- |
|  |

Figura 14 - Processador IO2I (teórico)

**Referências**

Wolf, Marilyn. *High performance embedded computing: architectures, applications, and methodologies*, 2nd edition. Elsevier Inc., 2014. ISBN 978-0-12-410511-9.

**APÊNDICE A — Diagrama completo dos Processadores**

|  |
| --- |
|  |

Figura 15 - Processador I2OI implementado

|  |
| --- |
|  |

Figura 16 - Processador IO2I implementado