Centro Federal de Educação Tecnológica de Minas Gerais

Departamento de Computação
Curso de Graduação em Engenharia da Computação
Laboratório de Arquitetura e Organização de Computadores II
Profa. Poliana Aparecida Corrêa de Oliveira (poliana@cefetmg.br)

Prática 3 – Algoritmo Tomasulo
Data da Entrega: 27/10/2023 – 25 pontos

O objetivo desse projeto é usar a linguagem Verilog para implementar o algoritmo Tomasulo de despacho simples sem especulação como descrito no livro texto da disciplina teórica (Seções 3.4 e 3.5).

HENNESSY, John L.; PATTERSON, David A. **Computer architecture: a quantitative approach**. 5 ed. Elsevier, 2011.

Descrição

No algoritmo Tomasulo a execução das instruções é dividida em 3 estágios: despacho, execução e *write back*. Esses três estágios acessam componentes críticos de hardware: o CDB, as estações de reserva (nas quais ocorrem as renomeações) e as unidades funcionais. Você deverá implementar:

(1) as estações de reserva;
(2) os estágios do algoritmo;
(3) as unidades funcionais de: **inteiro (soma/subtração), load e store**;
(4) o banco de registradores;
(5) as interconexões (barramentos e fios) e
(6) a fila das instruções.

Preparação

Inicie seu projeto criando todos os blocos básicos. Podem ser utilizados componentes da LPM. Caso seja possível, você também pode usar blocos criados para a Prática 2, desde que sejam de sua autoria, não é permitido usar código de outros grupos. Nesta versão simplificada do Tomasulo, somente serão realizadas operações de soma, subtração, leitura e escrita em memória. Desta forma, inicialize o banco de registradores e as primeiras posições de memória. Implemente uma arquitetura com 2 registradores.

A unidade funcional de inteiro requer 3 ciclos de execução.

Descrição **INICIAL** dos blocos básicos:

- **CDB arbiter**: arbitra qual unidade funcional poderá colocar o dado no CDB e faz o *broadcast* do resultado para o resto das unidades funcionais e o banco de registradores.
- **CDB**: barramento comum de dados que fornece o valor e o rótulo da estação de reserva que produz o dado.
- **Estações de reserva**: armazenam as instruções. Você poderá criar um componente separado para armazenar os registradores renomeados (ex, como uma tabela de renomeações).
- **Unidade funcional**: realizam as operações Soma/Subtração do tipo-R. Podem ser criadas unidades funcionais de inteiro separadas para instruções do tipo-R e tipo-I. Nesse caso, não será necessário implementar instruções do tipo-I.
- **Memória de dados e instruções:** armazenam dados nas primeiras 5 posições e instruções nas posições seguintes.
- **Fila de instruções**: Buffer que contém as instruções.
- **Banco de registradores**: Conjunto de registradores que armazenam os dados.

Esta é apenas uma descrição inicial dos componentes. Você deverá ser refinar, apresentar os detalhes e todas as decisões tomadas para criar o seu projeto.

Por onde começar?

Pense na arquitetura do seu processador em alto nível (projeto *top-down*): quais componentes são necessários, como contectar esses componentes, ou seja, quais sinais são necessários. Em seguida, detalhe as funções de cada um dos componentes. Depois, implemente e simule individualmente cada um deles a fim de verificar o

correto funcionamento. Por fim, faça a integração de todos os módulos e simule a versão simplificada do algoritmo de Tomasulo.

Você deverá testar a sua implementação utilizando um código de testes, explicar no relatório as formas de onda para todos os casos de testes e submeter o seu projeto no Moodle.

Código de Testes

O código abaixo é uma versão inicial para testes da versão simplificada do algoritmo de Tomasulo. Se necessário, você deve **adicionar** novas instruções para demonstrar o funcionamento da solução proposta para os seguintes casos de teste:

a) Dependência de dados verdadeira ou RAW (*Read After Write*);

b) Dependência de saída ou WAW (W*rite After Write*);

c) Antidependência ou WAR (*Write After Read*);

d) Dependência/hazard estrutural (STALL - unidade funcional cheia);

e) Conflito no CDB;

f) Adiantamento na estação de reserva;

g) Renomeação de registradores.

SOMA e SUB com os mesmos operandos

INSTRUÇÃO	R5	R4	R3	R2	R1	R0	MEM
	0	5	4	3	2	1	
ADD R1, R0	0	5	4	3	**3**	1	
ADD R0, R1	0	5	4	3	3	**4**	
ST R2, [R4]	0	5	4	3	3	4	**MEM[0] = 3**
SUB R0, R2	0	5	4	3	3	**1**	
LD R0, [R3]	0	5	4	3	3	**9**	MEM[4] = 9
ST R4, [R3]	0	5	4	3	3	9	**MEM[4] = 5**
SUB R0, R1	0	5	4	3	3	**6**	

Submissão

Crie um pacote contendo TODOS os códigos fontes, formas de onda, e o relatório do projeto. O nome do arquivo deve ser **"Pratica3_nomedoaluno1_nomedoaluno2.zip"**. Cada grupo deverá submeter apenas um pacote no Moodle.

O relatório deverá incluir os seguintes componentes:

1. Uma **introdução** em alto nível da sua solução para o algoritmo Tomasulo (por favor, não é para copiar a descrição do livro texto 😊).
2. O **projeto** do seu processador, incluindo detalhes necessários dos módulos criados.
3. Faça uma **figura** mostrando os blocos básicos e interconexões.
4. O código de **teste** utilizado e as formas de onda **com uma explicação que mostre o correto funcionamento**.
5. **Dificuldades** encontradas.
6. **Sugestões** de melhorias da prática.
7. **Comentários** adicionais.

Apresentação em sala

Cada grupo deverá apresentar a estrutura do código e mostrar os testes realizados (simulações) que comprovem o correto funcionamento da solução proposta para os seguintes casos de testes: a) Dependência de dados verdadeira ou RAW (Read After Write); b) Dependência de saída ou WAW (Write After Write); c)Antidependência ou WAR (Write After Read); d) Dependência/hazard estrutural (STALL - unidade funcional cheia); e)Conflito no CDB; f) Adiantamento na estação de reserva; g) Renomeação de registradores.

Não é necessário a apresentação na placa.

Pontuação

- Código: 30%
- Testes (simulações): 50%
- Relatório: 20%

Pontos Extras

- Inclusão da instrução de desvio e dos demais componentes necessários para o funcionamento – 2 pontos
- Apresentação na placa – 6 pontos
- Despacho duplo das instruções – 6 pontos
- Tomasulo completo com especulação – 12 pontos