Centro Federal de Educação Tecnológica
de Minas Gerais

Laboratório de arquitetura e organização de
computadores 2

Prática 03 – Implementação do Tomasulo

Renan Siman Claudino
Rubio Torres Castro Viana

Orientado por
Prof. Ms. Poliana Aparecida Corrêa de Oliveira

Belo Horizonte - MG
2018

Conteúdo

1 Introdução

Melhorar o desempenho é uma busca na área da computação desde que essa foi inventada[1]. O algoritmo tomasulo nos permite paralelizar instruções através do gerenciamento de conflitos de dados. Foi aplicado nesse projeto o tomasulo básico com quatro instruções: add, sub, mult e div, com intuito de demonstrar o seu funcionamento.

2 Detalhes do projeto

2.1 Decisões de implementação

Para esse projeto optamos por usar instruções de nove bits, sete registradores, sendo que cada registrador possui nove bits.
Nossas instruções possuem o formato:

RegX	=	RegX "operador"RegY

Os nove bits da instrução estão divididos em:

RegX	RegY	Instrução
8 - 6	5 - 3	2 - 0
000	000	000

E os códigos das quatro instruções estão dispostos como:

	RegX	RegY	Instrução
	8 - 6	5 - 3	2 - 0
ADD	000	000	000
SUB	000	000	001
MULT	000	000	010
DIV	000	000	011

2.2 Módulos

Foram implementados 8 módulos principais, parte deles para o funcionamento do tomasulo, parte para as simulações e parte para a apresentação em sala:

- Unidade funcional de soma e subtração: Realiza operações de soma e subtração.
  Neste módulo, além das operações, possui um contador para verificar a latência de 2 ciclos;

- Unidade funcional de multiplicação e divisão: Realiza operações de multiplicação e divisão;
  Neste módulo, além das operações, possui um contador para verificar a latência de 3 ciclos para multiplicação e 4 ciclos para divisão;

- Fila de instruções: Gerencia os despachos para estação de reserva;
  Neste módulo implementamos as instruções pré escritas na memória, verificando se as estações de reserva (addsubfull, muldivfull) estão cheias. Caso não estejam, liberamos uma nova instrução;

- Registrador de dados: Gerencia os registradores;

- Sete segmentos: Gerencia a impressão em display de sete segmentos;

- Contador: Gerencia o tempo do tomasulo;

- Controle e estação de reserva: Implementa o CDB, o controle, a estação de reserva e o controle de despachos;
  A estação possui seis posições, sendo as três reservadas para Adição e Subtração e as outras três para Multiplicação e Divisão. A estação armazena os valores e tags I e J, Instrução, Tempo de Escrita, busy e Execução;
  Para a escrita de dados implementamos a escrita por tempo. A operação de menor tempo é escrita no CDB, além de atualizar as TAGS após as instruções;

- Tomasulo: Implementação na placa, auxilia nas simulações;

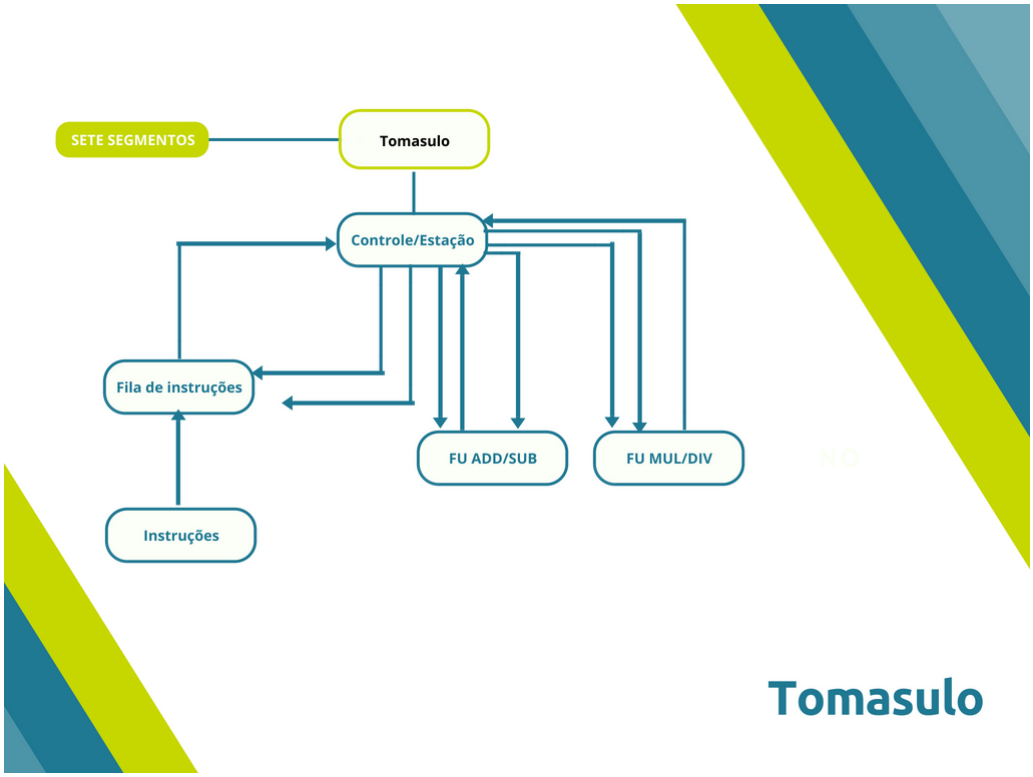

Figura 1: Projeto Tomasulo

3 Funcionamento

Verificamos se há espaço na estação de reserva. Caso haja, a fila libera uma instrução que é colocada na estação. Após isso verifica-se dependência de dados verdadeira, se houver é setado o label e deixado o valor como "NULL", se não houver carrega-se o valor e zera a tag. Se houver unidades funcionais livres a instrução é executada e os labels dependentes dela zerados. Se houver duas unidades funcionais liberadas ao mesmo tempo para escrever no CDB, escreve primeiro a que tiver menor tempo.

4 Testes

Foi realizado testes com 2 sequências de códigos diferentes: O primeiro código testamos todos os hazards, iniciando os registradores com dois e com a sequência de código:

Instrução
Regs2 = Regs2 + Regs1
Regs2 = Regs2 - Regs1
Regs1 = Regs1 + Regs0
Regs1 = Regs1 + Regs0
Regs1 = Regs1 - Regs0
Regs0 = Regs0 + Regs0
Regs0 = Regs0 - Regs1

E foi obtido o seguinte resultado:

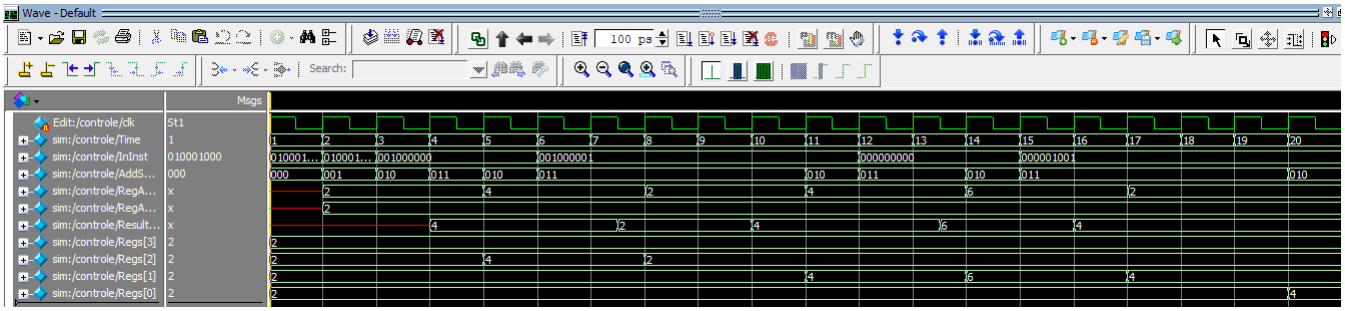

Figura 2: Testes Hazard

É possível ver um atraso por dependência de dados, um atraso por Unidade funcional já sendo usada, e um atraso por estação de reservas cheia. De verde podemos ver o stall por

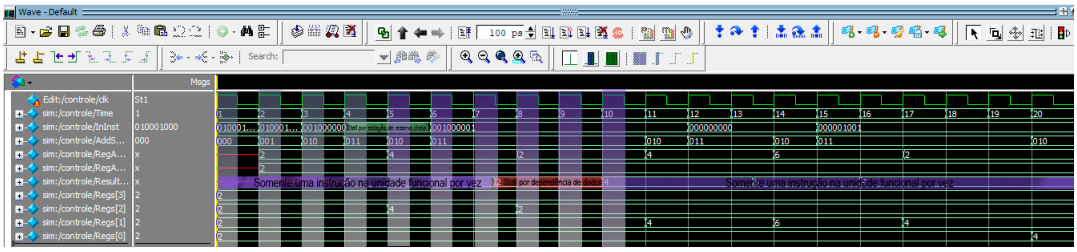

Figura 3: Teste Add

estação de reserva cheia provocada por mais de 3 instruções sendo colocadas. De azul temos um único barramento de saída para cada unidade funcional para que assim somente um instrução seja executada a cada unidade funcional. De vermelho vemos um dos stalls por dependência de dados.

Após isso fizemos um teste com instruções mistas:

Instrução	Despacho	Exe/Mem	CDB	Comentário	Conteúdo
Reg0 = Reg0 + Reg1	1	2 - 3	4	1º despacho	R0 = 2
Reg0 = Reg0 * Reg0	2	5 - 7	8	Espera add	R0 = 4
Reg1 = Reg1 - Reg2	3	8 - 9	10	Espera FU	R1 = 0
Reg2 = Reg2 * Reg1	4	10 - 12	13	Espera FU e SUB	R2 = 0
Reg3 = Reg3 + Reg3	5	10 - 11	12	Espera FU	R3 = 2
Reg0 = Reg0 + Reg0	6	14 - 15	16	Espera mul	R0 = 8
Reg0 = Reg0 / Reg3	7	17 - 18	19	Espera add	R0 = 4

E obtivemos esse resultado:

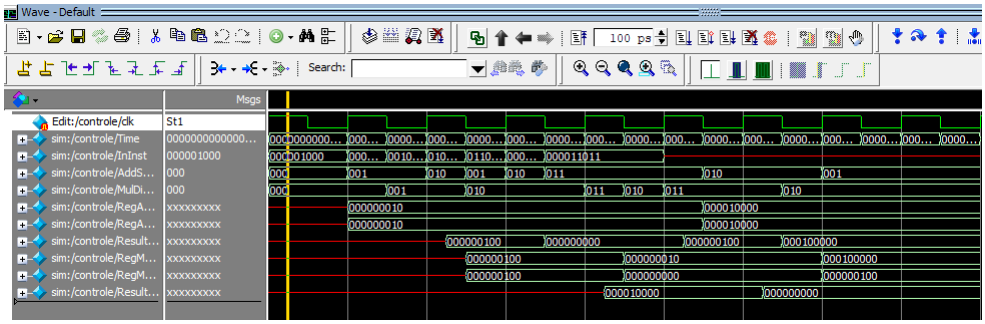

Figura 4: Testes misto

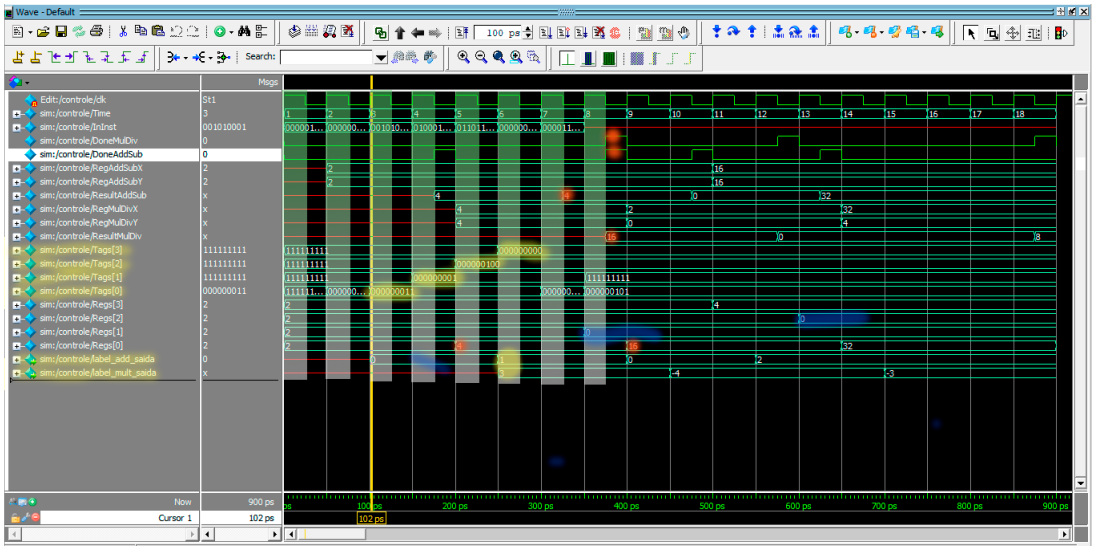

Figura 5: Testes misto

Em vermelho é possível mostrar a escrita no cdb quando as duas unidades funcionais dão resultado ao mesmo tempo, para tal comparamos o tempo das duas instruções o de menor tempo é escrito no cdb. Em amarelo é possível ver as tags e o labels que são trocados ao ter alguma dependência para a realizar a renomeação. Em azul mostramos um caso de (WAR).

5 Dificuldades

Houve problemas de comunicação ao usar os módulos CDB e estação de reserva separados. Tentamos implementar a LPM para despacho de instruções mas devido o delay, não obtivemos sucesso. A linguagem verilog não ajudou muito, pois suas limitações nos complicam ao pensarmos no projeto.

6 Sugestão

O tomasulo é uma implementação que junto ao verilog se mostra muito sensível a pequenas falhas, o que dificulta bastante pois um pequeno erro pode deixar travado por semanas. Seria interessante disponibilizar um esqueleto para guiar os alunos mais perdidos (como nós).

7 Conclusão

Foi possível implementar o algoritmo tomasulo com sucesso, entretanto apareceram erros como a discrepância entre o tempo do tomasulo feito "em mãos"e o do algoritmo e alguns bugs aleatórios que aparecem as vezes em algumas simulações. Aproveitamos para agradecer aos veteranos que se disponibilizaram ajudar em alguns bugs e ao monitor.

Referências

[1] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.