



PUC Minas

ALGORITMO DE TOMASULO

SIMULADOR DE ARQUITETURA

Apresentado por:

Pedro Heinrich Sales Pena
Raul da Cruz Fonseca

Objetivo e motivação

- Reproduzir a dinâmica do algoritmo de Tomasulo para estudar renomeação, forwarding e commit fora de ordem.
- Ferramenta com GUI para visualizar RS, ROB, registradores e métricas em tempo real.
- Código modular em C++ com testes unitários (GoogleTest) e build com CMake.



Figura 1. Tela principal. Autoria própria, 2025.

Tomasulo

O algoritmo de Tomasulo busca **maximizar o paralelismo** e **resolver conflitos da execução fora de ordem** por meio de **renomeação de registradores**, que elimina dependências falsas; estações de reserva, onde as instruções aguardam operandos; o Common Data Bus, que difunde resultados para todas as unidades dependentes; e o **Reorder Buffer**, que finaliza as instruções na ordem correta e mantém a consistência do programa.

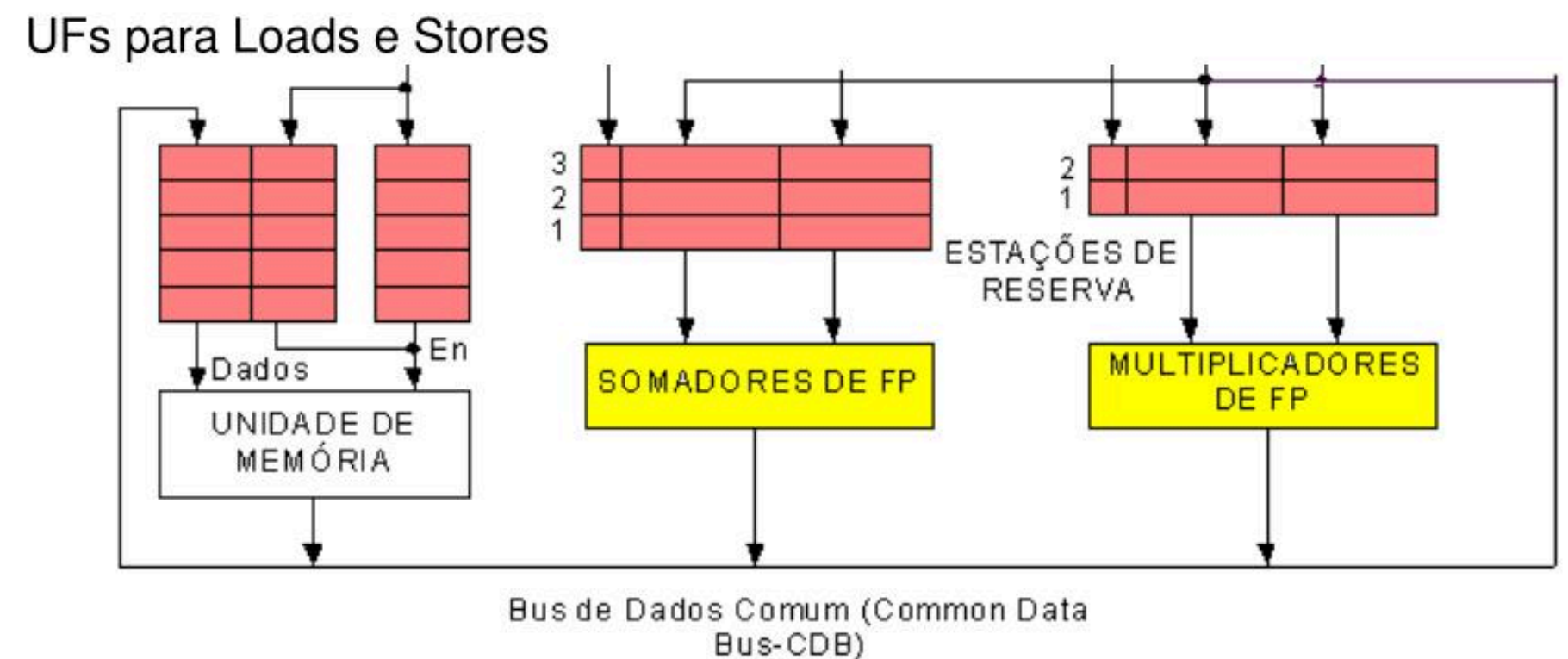


Figura 2. Diagrama de execução. DAP Spr, 1998.

Pipeline

- **Issue**
 - Despacha instruções e renomeia registradores
- **Execute**
 - Monitora operandos e executa cálculos
- **Write**
 - Escreve resultados no CDB e no ROB
- **Commit**
 - Efetiva os resultados na ordem

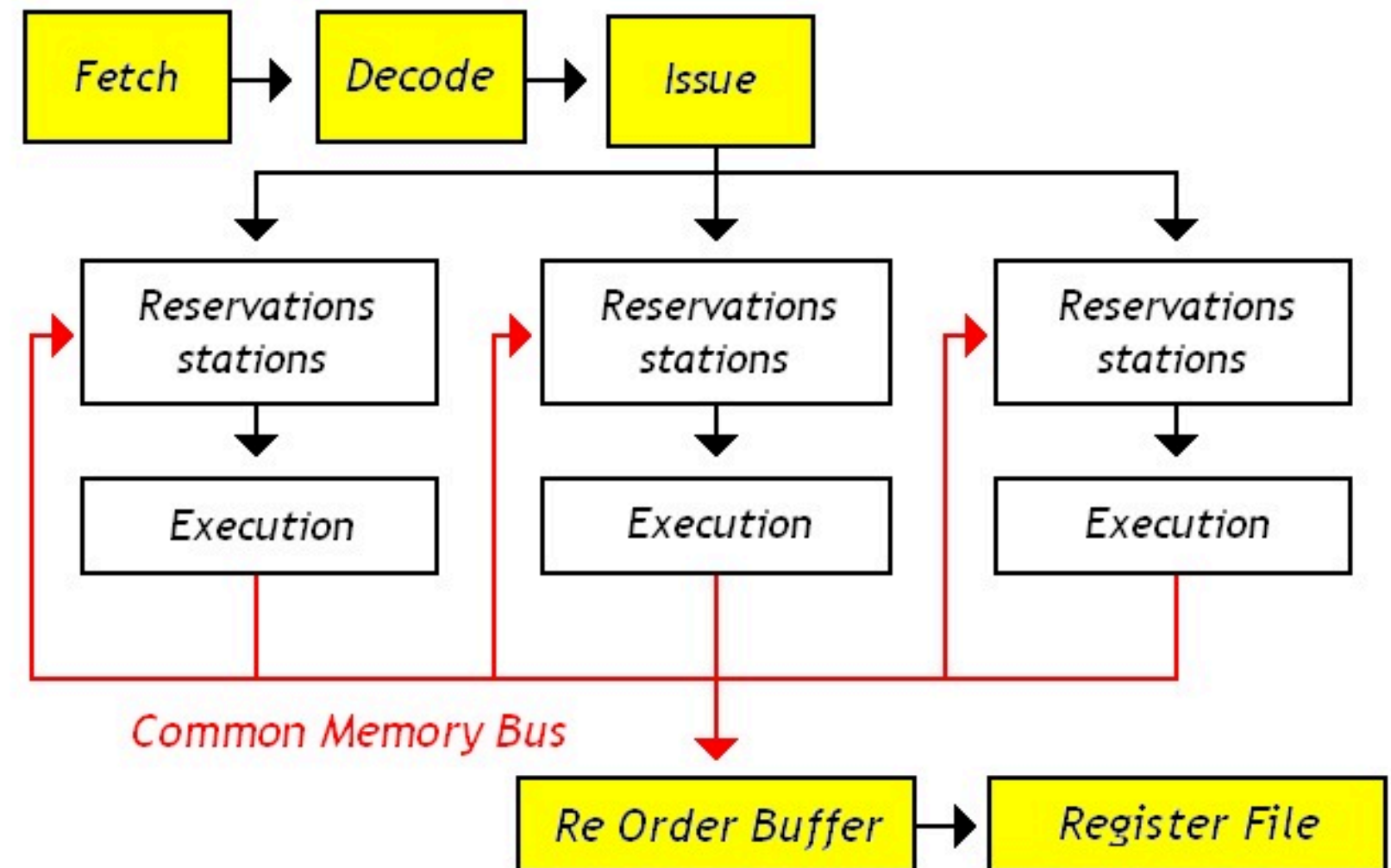
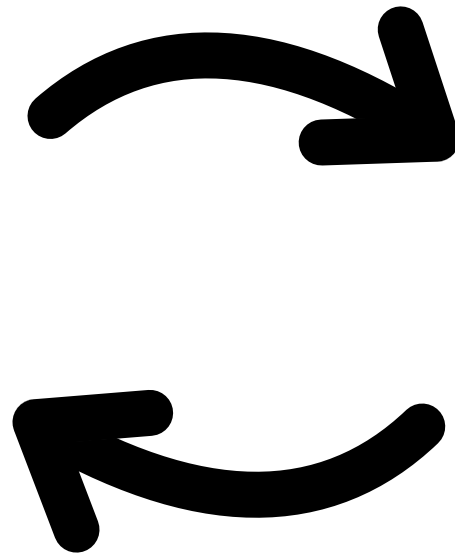


Figura 3. Diagrama de pipeline. Stafford Horne, 2019.

Arquitetura de projeto

tomasulo_core.*

- reservation_station.*
- reorder_buffer.*
- common_data_bus.*
- register_file.*
- instruction.*



C++ e QT6

Detalhes de implementação

reservation_station.*

Entradas com campos (busy, op, Vj/Vk, Qj/Qk, robIndex, cyclesLeft).

register_file.*

Mantém produtor (tag) por registrador e atualiza no commit.

common_data_bus.*

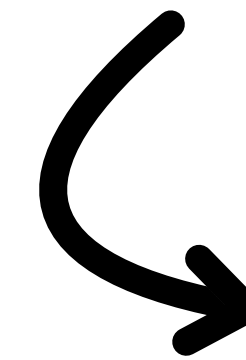
broadcast de (tag, value) para unidades que aguardam

reorder_buffer.*

Buffer circular que controla commit e liberação de recursos

instruction.*

Declara/implementa a estrutura que representa cada instrução



Branches

JAL tratado no issue; BEQ/BNE são avaliados no commit e removem entradas mais jovens.

Detalhes de implementação

- Renomeação via tags do ROB (cada alocação no ROB retorna um tag usado como producer)
- Forwarding implementado via `rob.isReady()` e snoop do CDB nas RS e ROB
- Flush: remove entradas do ROB e limpa mapeamentos do RAT / RS que dependem de tags mais jovens

Arquivo Exibir Ajuda

Executar Pausar Passo Resetar

Ciclo: 8

Métricas de desempenho

IPC médio: 0.50
Ciclos: 8
Instruções confirmadas: 4
Ciclos de bolha: 4
Estado: Pausado

Pipeline Estações de Reserva **Reorder Buffer** Registradores Logs Preferências

Idx	Busy	Op	Dest	Valor	Estado
0	Não	ADDI	R1	-	Writeback
1	Não	ADDI	R2	-	Writeback
2	Não	ADDI	R3	-	Writeback
3	Não	ADD	R4	-	Writeback
4	Sim	MUL	R5	0.00	Issue
5	Sim	SUB	R6	0.00	Issue
6	Sim	ADD	R7	0.00	Issue
7	Sim	SW		0.00	Issue

Figura 4. ROB durante execução. Autoria própria, 2025.



DEMONSTRAÇÃO



Conclusões

O projeto resultou em um simulador funcional do algoritmo de Tomasulo, acompanhado por uma interface gráfica e métricas em tempo real, permitindo visualizar renomeação de registradores, forwarding e o impacto direto das latências das unidades funcionais na execução. Apesar disso, o modelo ainda adota simplificações, como uma abstração de memória baseada em mapa e a ausência de um mecanismo avançado de predição de desvios, além de utilizar contadores e métricas essenciais que podem ser expandidos.

MOMENTO DE PERGUNTAS

