Relatório do Trabalho Prático 2
Arquitetura de Computadores

Gabriel Lopes Ferreira
Luiz Junio Veloso Dos Santos
Matheus Luiz Oliveira Spindula
Rebeca Neto

31 de outubro de 2019

1. **Introdução:** O objetivo
2. **O simulador:**
3. **Análises:**
   (a) **Teste 1**
   Neste teste, usamos uma arquitetura superescalar com *2-way Pipeline*, *2-line buffer*, um trace com 21 instruções, com todas as instruções idênticas (’ADD R1,R2,R3’), ou seja, todas leem e escrevem no mesmo local, comparamos o impacto de utilizar 1 ALU vs 2 ALU.

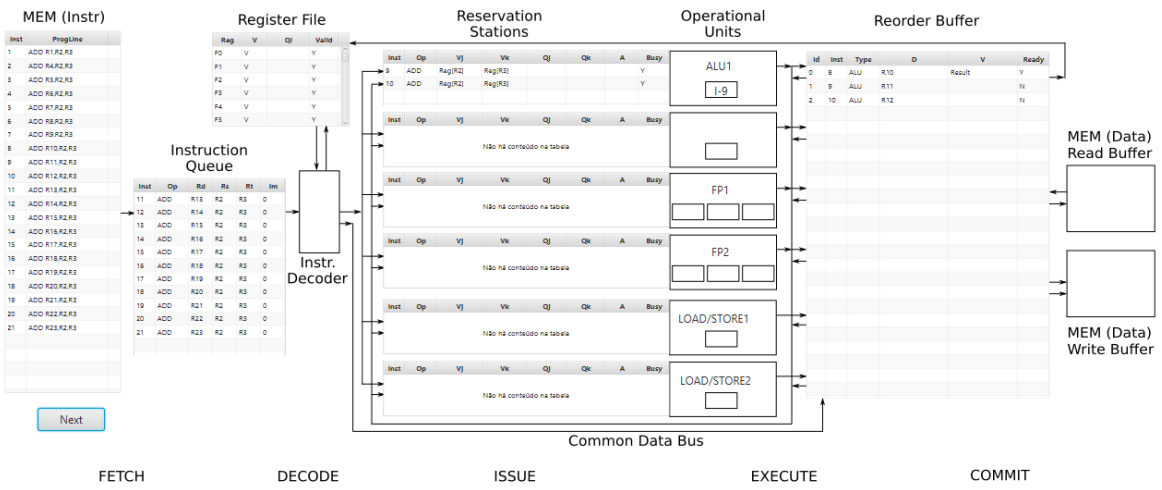

Figura 1: Execução com 1 ALU. Total de Ciclos no teste: 26

No teste acima, apesar de muitas instruções serem despachadas para o Instruction Queue, poucas vão para o *Reservation Station* (Janela Distribuida), já que existe somente 1 ALU e sua janela comporta somente 2 instruções e somente uma é executada em cada ciclo. Após sair da ALU o resultado vai para o *Reorder Buffer*, para que possa ser escrito no banco de registradores. Como neste trace todas as instruções escrevem no mesmo local, existe um “acumulo” de instruções no *buffer* com resultado pronto, porém aguardando uma instrução anterior terminar a escrita.

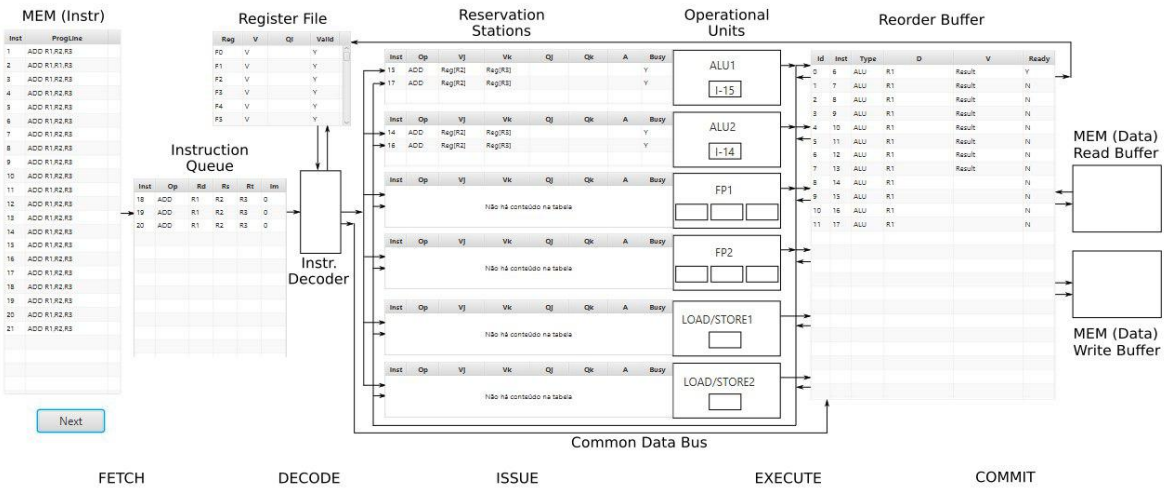

Figura 2: Execução com 2 ALU. Total de Ciclos no teste: 25

Na segunda parte do teste, foi adicionada uma segunda ALU, isso fez com que um maior número de instruções fossem despachadas para as *Reservation Stations*, gerando um menor acumulo de instruções no *Instruction Queue*, o fato de ter 2 ALU's faz com que ocorra um paralelismo de instruções, duas em cada janela. Contudo esse paralelismo fez com que houvesse um maior acumulo de instruções no *Reorder Buffer*.

(b) **Teste 2**

Neste teste, usamos uma arquitetura com *2-way Pipeline*, *2-line Buffer* e um trace com 21 instruções, todas fazendo uso da ALU ('ADD RX,R2,R3'), porém diferentemente do teste 1, todas as instruções escrevem em local diferente. Comparamos o impacto ao utilizar 1 ALU e 2 ALU.

No teste acima, vemos muitas instruções paradas no *instruction queue* aguardando serem despachadas, já que existe somente 1 ALU e sua janela há espaço para duas instruções. Já no *instruction queue* vemos somente 3 instruções, uma que está escrevendo no banco e outras duas que estão aguardando o resultado da ALU.

Nessa parte do teste, foi adicionada outra ALU, que impactou de forma bem significativa o desempenho do trace. Na imagem acima podemos ver somente duas instruções aguardando no *Instruction Queue*, e 4 instruções na etapa de execução, duas em cada ALU. Já no *Reorder Buffer* vemos um número maior de instruções em comparação ao teste anterior, devido ao maior número de instruções sendo executadas simultaneamente, além disso é possível ver que duas instruções escrevem no banco de registradores ao mesmo tempo.

(c) **Teste 3**

Neste teste, usamos uma arquitetura com *2-way Pipeline*, *2-line Buffer* e um trace com 21 instruções, todas fazendo uso da ALU (instruções do tipo ADD), todas as instruções apresentam dependências com a instrução anterior. Comparamos o impacto de utilizar 1 ALU e 2 ALU.

No teste acima, vemos um grande número de instruções paradas no *Instruction Queue* aguardando a liberação int para uma ALU, na ALU vemos as duas instruções paradas aguardando suas dependências terminarem de escrever no banco de registradores, já no *Reorder Buffer* vemos as duas instruções que estão na ALU e uma terceira realizando a

escrita.

Nesta parte do teste adicionamos uma segunda ALU, que não trouxe nenhum impacto devido as dependências, podemos ver um grande número de instruções no *Instruction Queue*, nas ALU's podemos ver 3 instruções paradas aguardando suas dependências e somente uma instrução sendo executada, já no *Reorder Buffer* vemos 4 instruções aguardando seus resultados para escreverem no banco.

(d) **Teste 4**

Neste teste, utilizamos *2-way Pipeline*, *2-line Buffer* e um trace com 21 instruções todas sendo um ADD de ponto flutuante, e escrevendo em mesmo local, sem dependências. Comparamos o impacto ao utilizar 1 FP e 2 FP.

No teste acima, vemos um grande número de instruções paradas no *Instruction Queue* aguardando a finalização de alguma instrução na FP1, já na FP2 podemos ver duas instruções sendo executadas, no *Reorder Buffer* vemos somente duas instruções aguardando a finalização para realizar a escrita.

Na imagem acima vemos os teste com duas unidades FP que trouxe um grande impacto para a execução do trace, podemos ver um número parecido de instrução na *Instruction Queue*, porém nas unidades de FP vemos duas instruções sendo executadas em cada uma das FP's. No *Reorder Buffer* vemos duas instruções a mais do que no teste com uma FP, e apenas uma instrução escrevendo no banco de registradores.

4. **Análise do simulador**

5. **Conclusões finais**