

Relatório pratica 3 – Algoritmo Tomasulo

Arthur Severo

Victor Le Roy

1. Introdução:

Este trabalho consiste na apresentação e construção do algoritmo Tomasulo sem especulação.

2. Desenvolvimento:

Para este projeto, optamos por utilizar instruções de 12 bits, sendo eles:

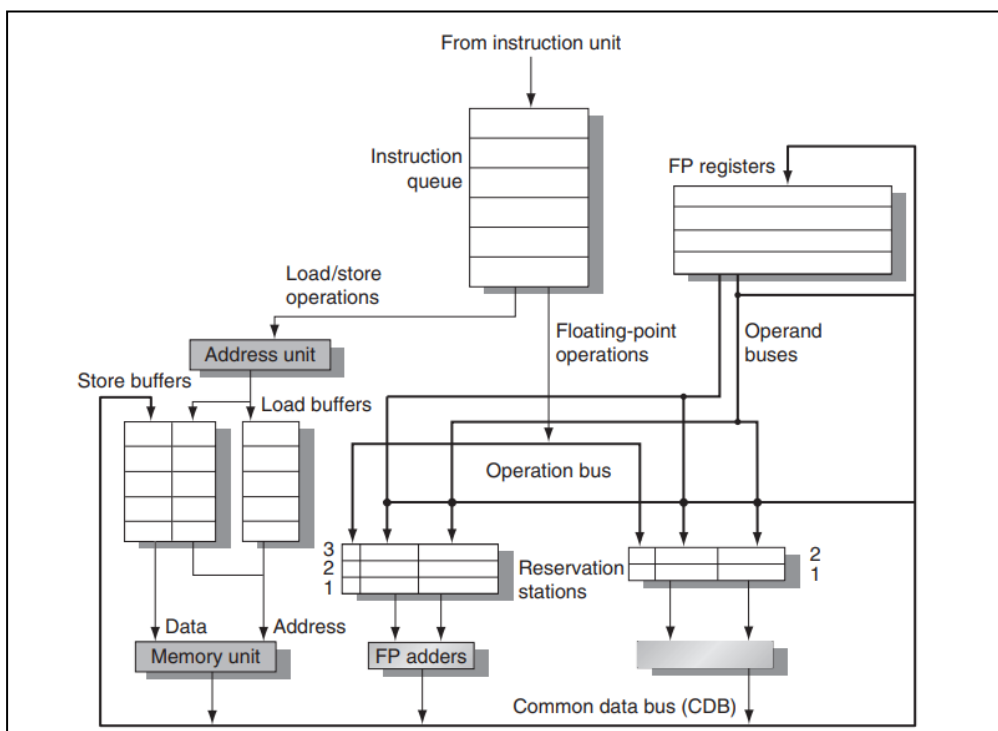
| 3 bits | 3 bits | 3 bits | 3 bits |
|--------|--------|--------|--------|
| Op | Rd | Rx | Ry |

Sendo Op, a operação, Rd, o registrador destino e Rx e Ry, os registradores fonte. Utilizamos como código de operação:

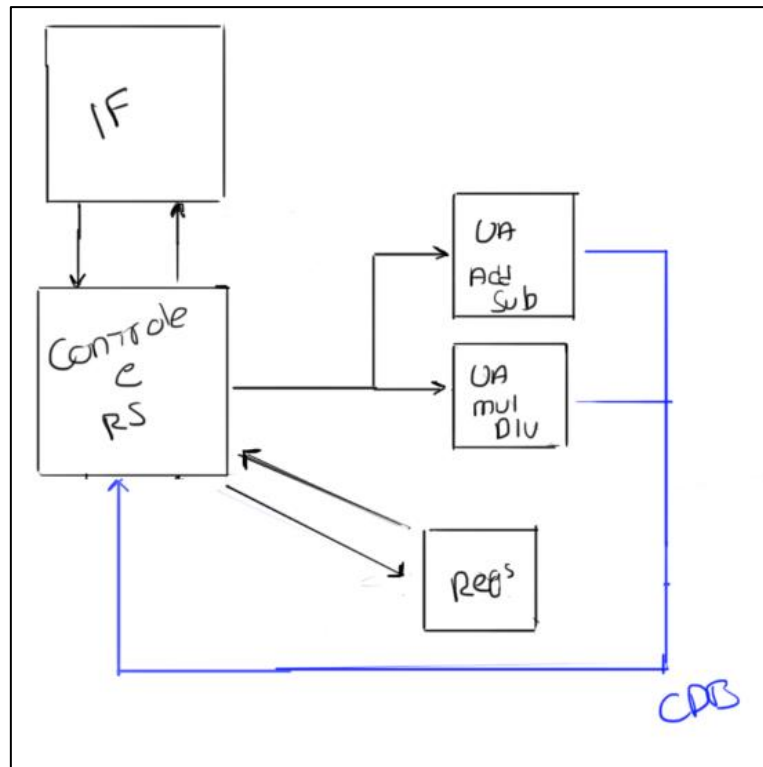
| Operação | Bits |
|----------|------|
| ADD | 000 |
| SUB | 001 |
| MUL | 010 |
| DIV | 011 |

3. Módulos:

Foi criado um diagrama a partir do esquema do algoritmo disponibilizado no livro da disciplina (referência 1), sendo este:



A partir do esquema anterior foi criado o seguinte diagrama:



- Unidade funcional/artimética (UA) AddSub:

Realiza operações de soma e subtração e possui a latência 1 ciclo.

- Unidade funcional/artimética (UA) MulDiv:

Realiza operações de multiplicação e divisão e possui a latência 2 ciclos.

- Fila de instruções (IF):

Gerencia os despachos da estação de reserva. Neste módulo, está setado uma memória contendo as instruções, além do contador do processador.

- Controle e RS:

Implementa o processador como um todo. Isto é, implementa o CDB, o controle de hazard e execução das instruções, além da estação de reserva.

4. Funcionamento

Primeiro verificamos se há espaço na estação de reserva. Caso haja, a instrução é despachada da fila e colocada no espaço vazio da estação. Após isso, verifica-se dependência de dados verdadeira, se houver é preenchido os valores e suas dependências. Se a instrução não tiver nenhuma dependência, esta é executada e no após a latência, é escrito o valor no registrador.

5. Testes

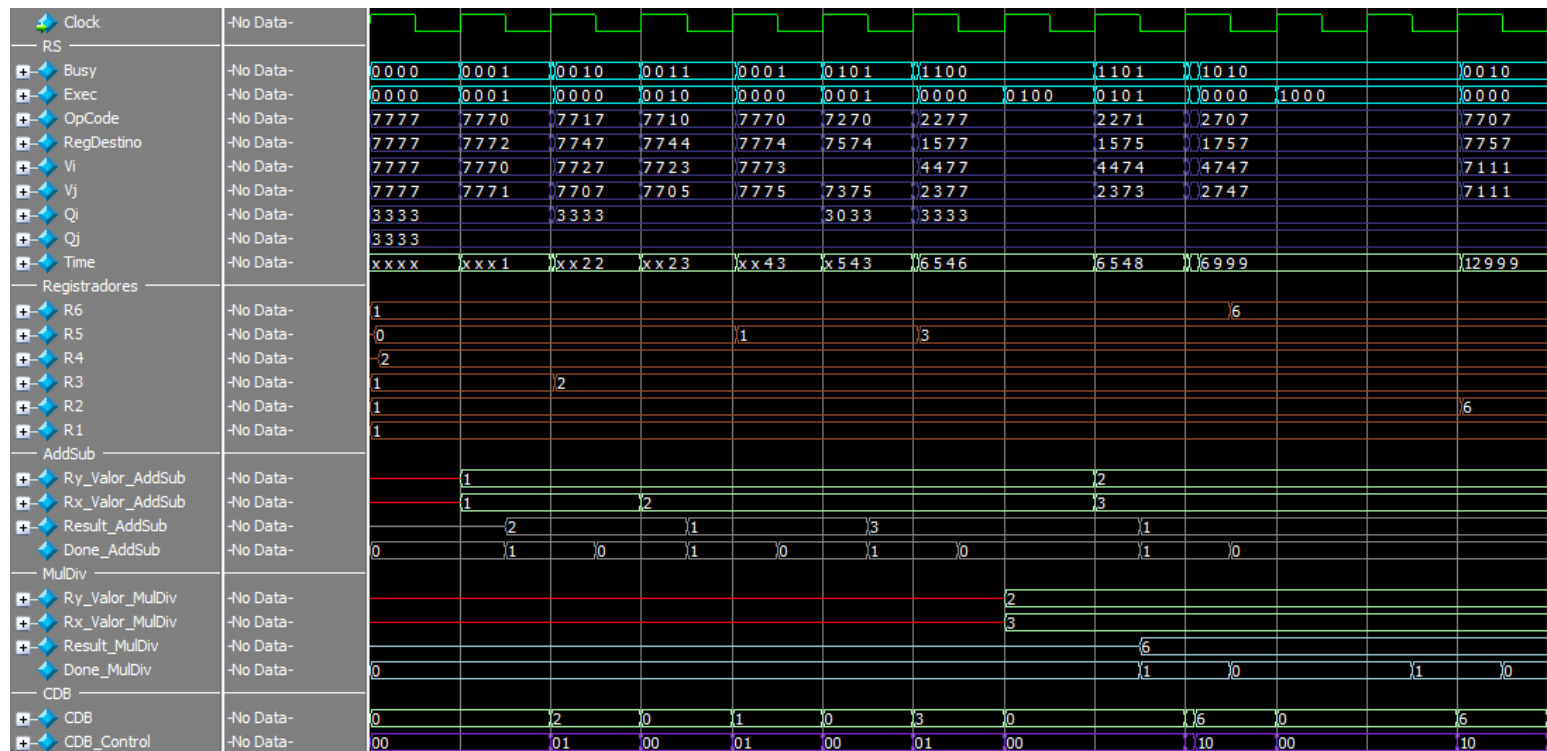
Foi realizado os testes a partir do código a seguir:

| Instrução/Ciclos que fica na estação de reserva | Emissão | Executa | Mem | Escreve CDB | Comentário |
|---|---------|------------|-----|-------------|--|
| ADD R3, R1, R2 [1-2] | 1 | 2 [soma] | - | 3 | Sem atraso. |
| SUB R5, R3, R1 [2-4] | 2 | 4[soma] | - | 5 | Mostrar dependência de dados verdadeira |
| ADD R5, R4, R6 [3-5] | 3 | 5 [soma] | - | 6 | Não há dependência, mas atrasa por hazard estrutural. FU cheia. Dependência de Saída |
| MUL R6, R5, R4 [4-8] | 4 | 7-8 [Mul] | - | 9 | Mostrar dependência de dados verdadeira |
| MUL R2, R5, R3 [5-10] | 5 | 9-10 [MUL] | - | 11 | Unidade funcional cheia. |
| SUB R6, R5, R4 [6-7] | 6 | 7 [Soma] | | 8 | Espera R5 |
| ADD R6, R5, R5 [7-8] | 7 | 8[Soma] | | 10 | CDB cheio |

E os valores que deveriam acontecer são:

| | R1 | R2 | R3 | R4 | R5 | R6 |
|-----------------------|----|----|----|----|----|----|
| Inicialização | 1 | 1 | 1 | 2 | 0 | 1 |
| ADD R3, R1, R2 [1-2] | 1 | 1 | 2 | 2 | 0 | 1 |
| SUB R5, R3, R1 [2-4] | 1 | 1 | 2 | 2 | 1 | 1 |
| ADD R5, R4, R6 [3-5] | 1 | 1 | 2 | 2 | 3 | 1 |
| MUL R6, R5, R4 [4-8] | 1 | 1 | 2 | 2 | 3 | 6 |
| MUL R2, R5, R3 [5-10] | 1 | 6 | 2 | 2 | 3 | 6 |
| SUB R6, R5, R4 [6-7] | 1 | 6 | 2 | 2 | 3 | 1 |
| ADD R6, R5, R5 [7-8] | 1 | 6 | 2 | 2 | 3 | 6 |

E foi obtido:



É possível perceber uma diferença de 2 ciclos comparado com o previsto. Isso se deu, pois, uma instrução teve sua execução adiada, o que resultou neste problema e no adiamento das instruções subsequentes. Este adiamento fez com que a instrução **SUB R6, R5, R4 [6-7]** tivesse sua execução terminada junto com **MUL R6, R5, R4 [4-8]**, porém ainda é possível ver que, pelo fato de a segunda instrução escrever após a primeira, temos o resultado correto no **R6**. Infelizmente, não conseguimos identificar a solução do problema citado acima.

Além desse problema, foi visto que, em algumas simulações, algumas operações tiveram algum comportamento parecido com o problema citado anteriormente.

6. Conclusão:

Este projeto foi com toda certeza o mais desafiador que tivemos até então. Creio que a maior dificuldade veio do fato da falta de conhecimento avançado na linguagem Verilog, o que gerou os problemas citados anteriormente, que foram apenas exemplos dos problemas encontrados, porém, este desafio fez com que pudéssemos aprender bastante sobre a teoria da disciplina, além de aprender mais sobre o algoritmo. Aproveito para agradecer aos veteranos que se disponibilizaram a nos ajudar a resolver este projeto.

7. Referencias:

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