Modificações no Processador Multi-ciclo

Guimarães, João Guilherme M.
joaog95@live.com

Muniz, Lucas L. R.
lucaslc01@hotmail.com

3 de julho de 2019

1 Objetivo da prática

A nona aula prática da disciplina de Laboratório de Arquitetura de Computadores I, teve como objetivo realizar modificações no código do processador multi-ciclo desenvolvido na aula anterior, estas mudanças são:

- Os registradores no formato da instrução passam a ter 4 bits;
- Mudança na relação entre os registradores de cada instrução e
- Mudança no *Opcode* de algumas instruções.

2 Descrição das Atividades

O aumento de um bit de endereçamento no formato da instrução, permitiu que **RF** tivesse um acréscimo de 8 registradores, passando assim de 8 para 16, e para manter o padrão de projeto, o registrador referente ao **PC** também foi alterado, estando agora na posição 15, a última posição de **RF**.

As alterações entre as relações dos registradores de cada instrução, acarretou na criação de uma nova variável de controle, *WriteReg*, possibilitando com que as instruções possam salvar seus dados em um dos 3 registradores, **X**, **Y** e **Z**, sendo que antes, a utilização do registrador **X** era obrigatória.

E por último, a mudança no *Opcode* das instruções, acarretou em mudanças simples no código de teste e na estrutura do módulo de controle.

A seguir um resumo das alterações:

- Expansão de **RF** de 8 para 16 registradores;
- Substituição do registrador referente ao **PC** para a posição 15;

- Criação da nova variável de controle *WriteReg* e
- Correção dos *Opcodes* das instruções.

3 Simulação

Após as modificações necessárias, iniciamos a simulação utilizando a instrução *copy input*, pois ela é responsável por inicializar os dados nos registradores, sendo assim, se torna presente nas simulações das demais instruções. Segue abaixo o código, a representação e a descrição breve da simulação.

```
// Copy Input
Mem[ 5'h0 ] = 16'hf000;
Mem[ 5'h1 ] = 16'h001a;
Mem[ 5'h2 ] = 16'hf100;
Mem[ 5'h3 ] = 16'h000f;
```

Figura 1: Código da simulação da instrução *Copy Input*

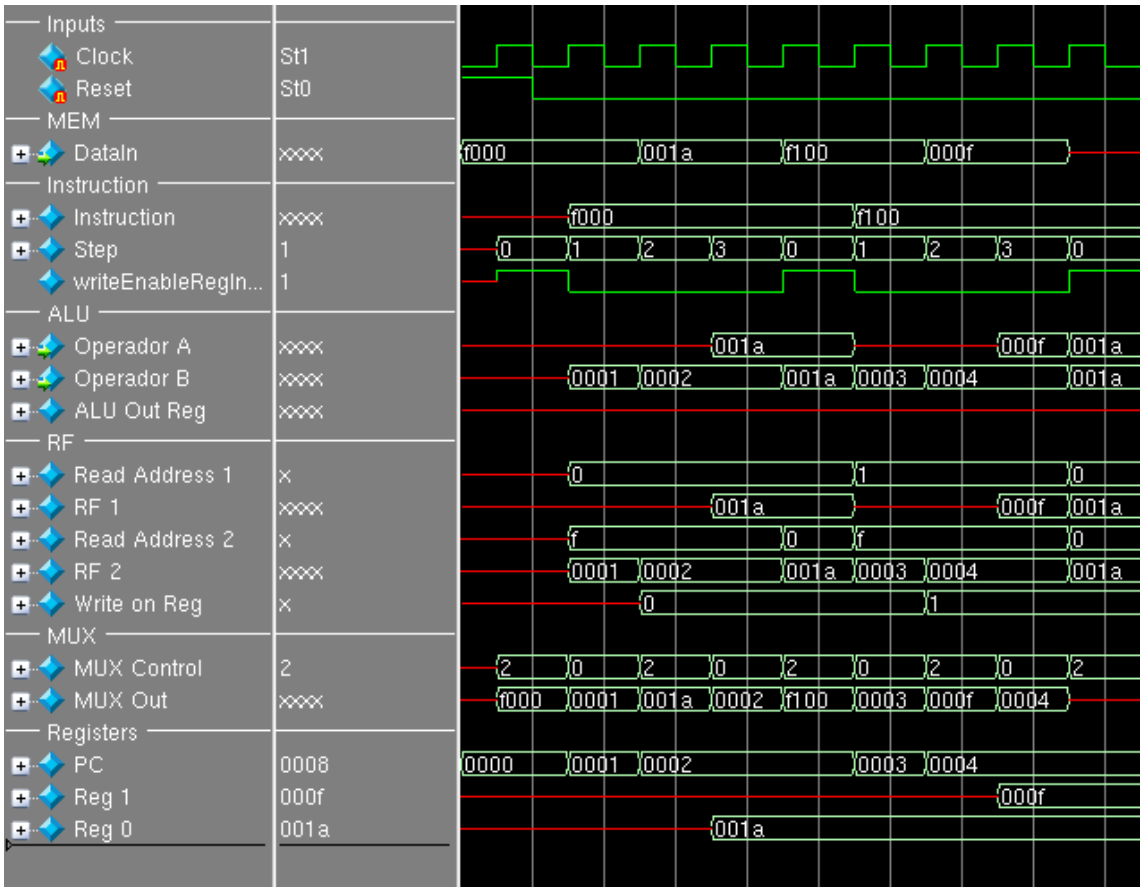

Figura 2: Simulação da instrução *Copy Input*

Baseando se na figura 2, temos a esquerda as entradas *Clock* e *Reset*, e as variáveis necessárias para visualizar melhor o *datapath* da instrução. A simulação consistiu em carregar da memória os dados *001a* e *000f* e salvá-los nos registradores *0* e *1*, respectivamente. O resultado da simulação ocorreu como esperado, já que no início do passo 3 de cada instrução, o registrador referente já estava escrito com o dado proposto.

A segunda instrução a ser simulada é a *not*, que consiste em realizar os mesmos procedimentos descritos na figura 1 para inicializar os registradores *0* e *1*, com acréscimo de inverter o dado no registrador **X** e armazena-lo no registrador **Y**.

```
// Not
Mem[ 5'h4 ] = 16'h0020;
```

Figura 3: Código da simulação da instrução *Not*

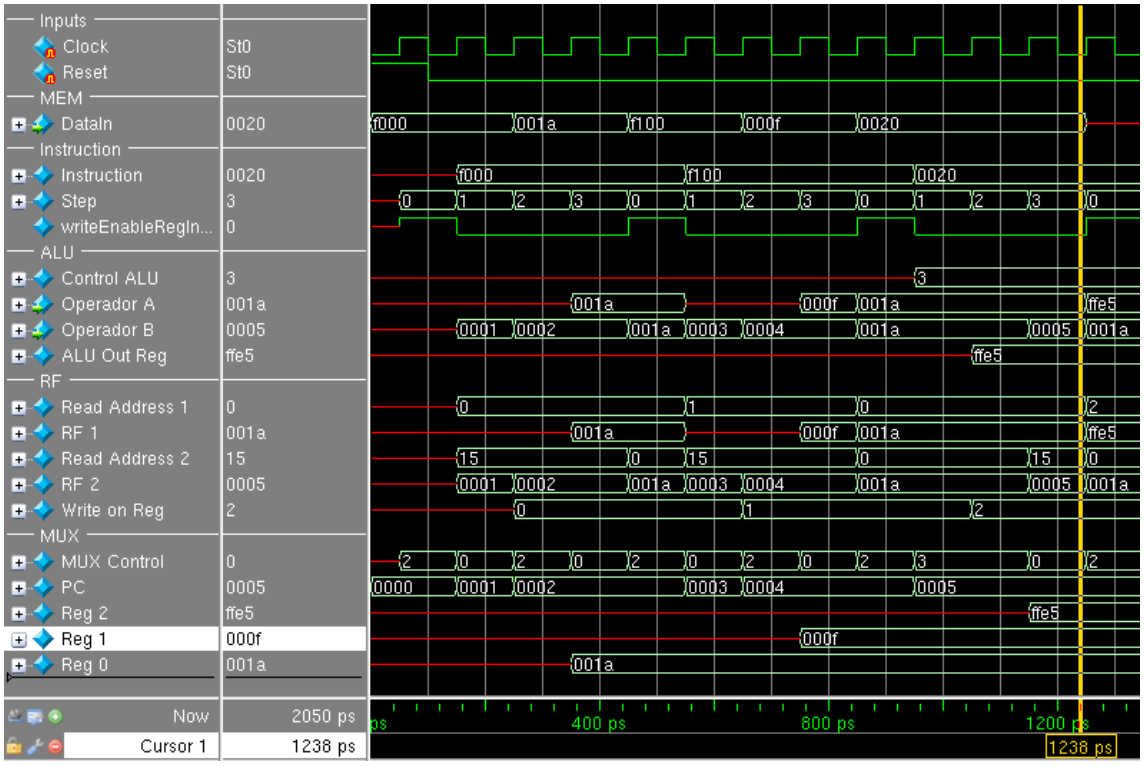

Figura 4: Simulação da instrução *Not*

Como pode ser visto na figura 4, o resultado obtido foi igual ao esperado, já que no terceiro passo da instrução *000f*, ocorreu a escrita no registrador *2*.

A última simulação descrita neste relatório, será a da instrução *store*, que novamente consiste em realizar os procedimentos descritos na figura 1 para inicializar os registradores e posteriormente, salvar na memória de endereço *11010 (001a)* o dado *000f*.

```
// Store
Mem[ 5'h4 ] = 16'hd010;
```

Figura 5: Código da simulação da instrução *Store*

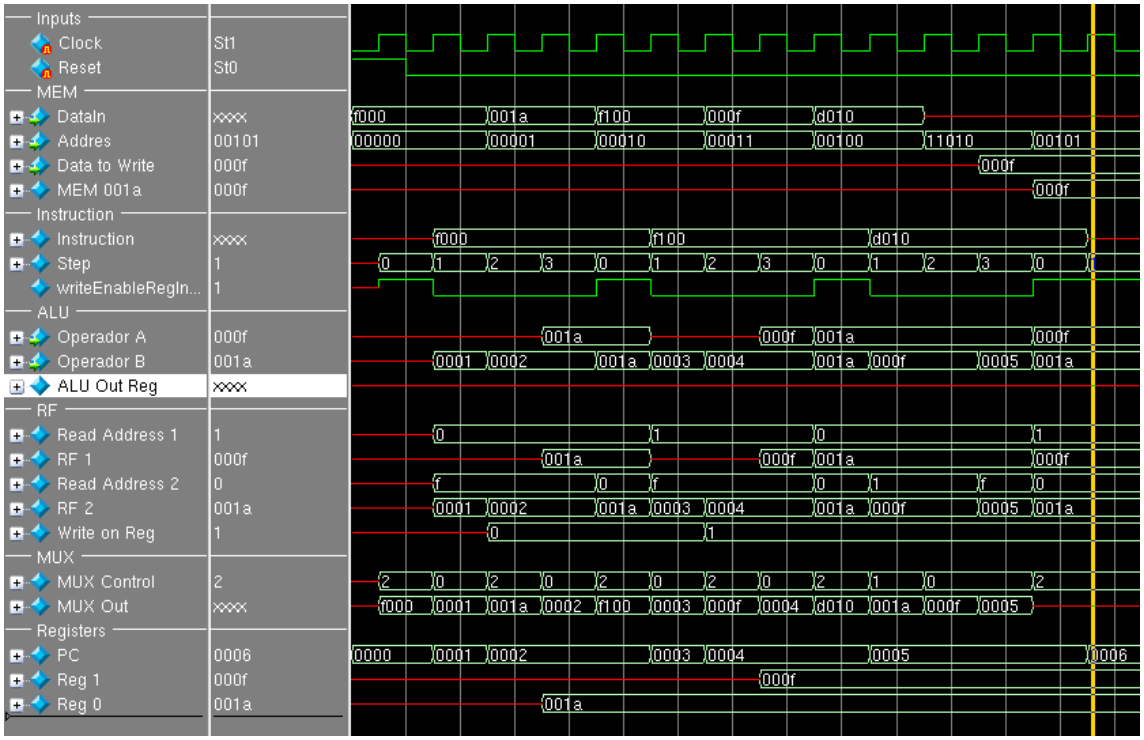

Figura 6: Simulação da instrução *Store*

Baseando se na figura 6, é possível perceber que a instrução *store* diferente das demais, tem o seu resultado obtido somente no passo 0 da próxima instrução, que foi de acordo com o esperado.

Obs.: o código e a simulação do restante das instruções se encontram na pasta do projeto, mais especificamente, na pasta simulação.

4 Dificuldades

As principais dificuldades obtidas no desenvolvimento desta prática foram:

1. Encontrar e resolver o erro na instrução *conditional copy*, já que a mesma realizava o inverso do proposto (executava o processo de *copy* somente quando o valor na **ULA** era diferente de 0) e

2. Realizar os testes de cada instrução após as alterações, o que resulta em um total de 9 simulações.

5 Conclusão

Com a execução desta prática, pudemos aperfeiçoar nossos conhecimentos em projetos de processadores multi-ciclos e da linguagem Verilog, além de exercitar o trabalho em equipe.