Universidade Federal Rural do Semiárido
Departamento de Computação
Ciência da Computação
Arquitetura e Organização de Computadores
Prof. Sílvio Fernandes

2.3 – Trabalho de Implementação

Trabalho em dupla ou individual
Data de entrega (via SIGAA): 11 de Setembro de 2023
O trabalho deve ser apresentado em sala de aula para o professor

Descrição:
Para este trabalho serão construídos o(s) caminho(s) de dados das instruções **ADD, SUB, AND, OR e SLT, BEQ e J**. O caminho de dados será montado a partir de implementações de componentes básico já projetados e fornecidos em um arquivo de projeto Logisim.

A definição completa do circuito do processador MIPS para o Logisim deve ser salvo com o nome dos componentes do grupo da seguinte forma: NomeA-NomeB-mips-datapth-TipoR-Saltos.circ.

Para começar:

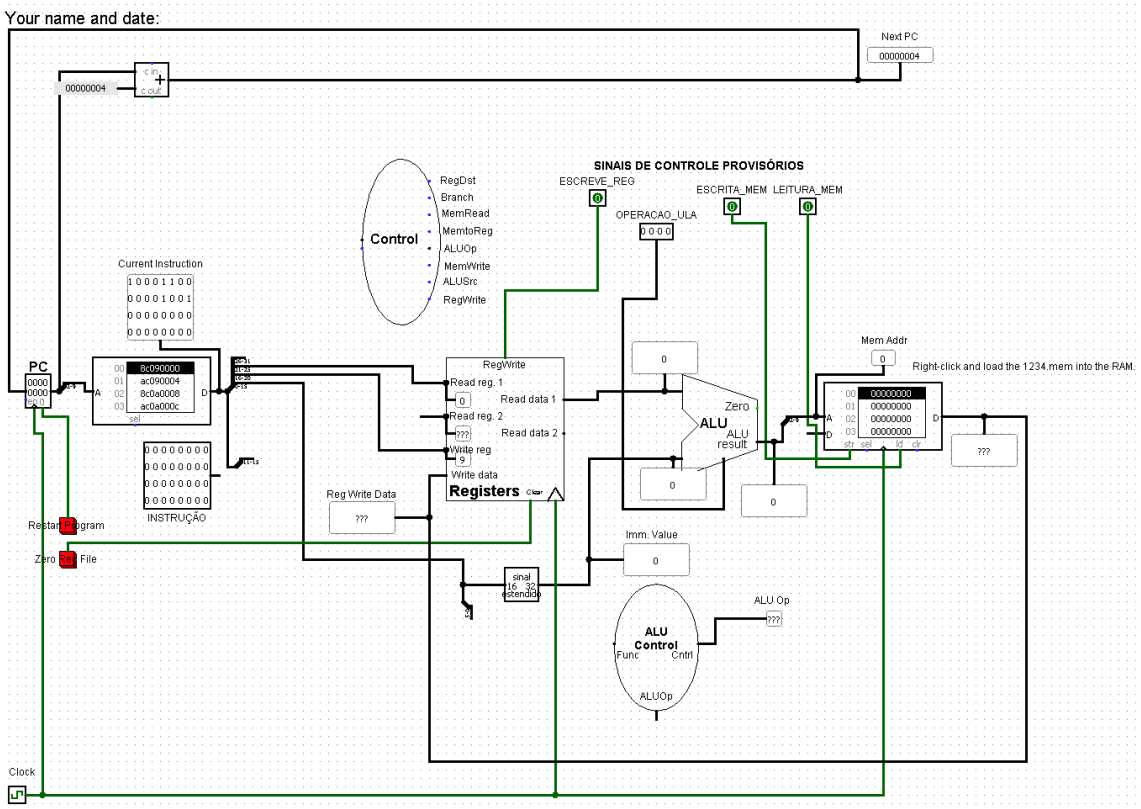
Leia o capítulo 4 do livro principal desta disciplina listado nas referências no final deste texto.

Utilize a versão 2.7.1 do Logisim. Para executar este software é necessário que em seu computador também esteja instalado o *Java Runtime*.

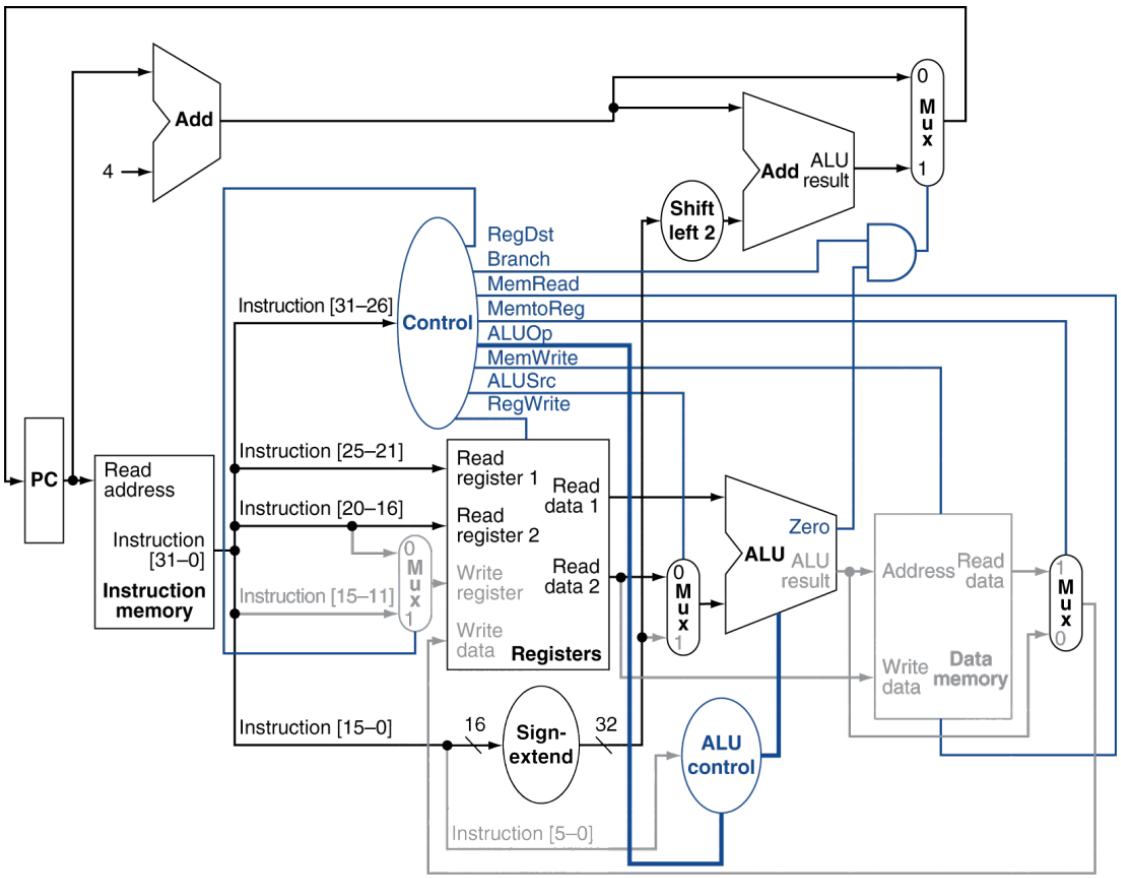
Utilize como ponto de partida o arquivo mips-datapath-lab.zip disponibilizado com esta definição do trabalho no SIGAA. Tal arquivo .zip contém os demais arquivos:

- mips-datapath-lab-R-Saltos.circ: um "esqueleto" do processador MIPS básico com importantes partes faltando propositalmente.
- 1234.mem: um arquivo hexadecimal que contém os valores 1, 2, 3, 4 para inicializar as primeiras 4 palavras da **memória de dados**.
- test-code.mem: um arquivo que contém um programa de teste simples para verificar seu projeto

Abra o arquivo de projeto Logisim ".circ" e verá o seguinte circuito correspondente ao MIPS com algumas partes faltantes:



Este circuito após completado deve ser equivalente ao processador apresentado no livro:



Implementação:

Passo 1: Ligando o caminho de dados

Abra o circuito “mips” do projeto. Preencham os **nomes dos componentes do grupo** que estão implementando o projeto.

Use a figura do livro e sua compreensão de como o processador MIPS funciona para concluir o caminho de dados incompleto que foi fornecido. Você não deve excluir nada que foi colocado no projeto disponibilizado. Ao longo do projeto já foram incluíndas várias “pontas de prova” que mostram os valores nos fios que entram e saem dos componentes para facilitar a depuração. Caso deseje, inclua nova “pontas de prova” para ajudar seu entendimento e depuração.

Tenha certeza de ligar os MUX conforme a figura do livro e incluir a outra lógica de atualização do PC. Para ter certeza de que está ligando os fios nas portas corretas dos componentes é recomendado dar um zoom (na parte inferior esquerda da janela) e passar o mouse sobre a porta para ver o nome dela.

Nesta atividade você deverá implementar as instruções do tipo R (**ADD, SUB, AND, OR e SLT)** e de salto (**BEQ e J**). Quando novos componentes forem adicionados, como por exemplo multiplexadores, é necessário incluir também sinais de controle para eles, inclua-os juntos com os demais sinais de controles provisórios do projeto fornecido.

Passo 2: Testando cada instrução individualmente

Despois de implementar o caminho de dados de cada instrução, testa-a individualmente. Para isso é necessário habililtar os sinais de controle adequadamente para que a instrução funcione.

Verifique se os valores correspondentes a cada campo da instrução estão sendo enviado para os componentes corretos e se as respostas calculadas por eles também estão corretas. Uma boa estratégia para isso é adicionar pontas de provas e escolher como gostaria de ver os dados do fio (decimal com sinal, sem sinal, binário ou hexa). Quando a instrução acessar registradores (leitura ou escrita) abra o banco de registradores para verificar se o acesso foi feito corretamente.

Passo 3: Testando o caminho de dados com um código

Para testar seu processador, fornecemos o seguinte programa:

```
lw  $t1,  0($zero)
lw  $t2,  4($zero)
lw  $t0,  12($zero)
loop:
add  $t3,  $t1,  $t2
add  $t4,  $t3,  $t1
sub  $t0,  $t0,  $t2
or   $t5,  $t4,  $t1
slt  $t6,  $t4,  $t5
beq  $t2,  $t0,  loop
```

Este programa aparece em código de máquina, na forma hexadecimal, no arquivo “test-code.mem” fornecido no projeto. Tal arquivo é usado para inicializar a memória de instruções.

E um arquivo de memória que contém os valores 1,2,3 e 4 nas primeiras 4 palavras da memória (1234.mem). Para usar esses arquivos, você precisa clicar com o botão direito do mouse na memória de instruções e nas memórias de dados e carregar os arquivos apropriados. Também é possível criar seus próprios programas e inicializar as memórias com as informações que desejar para executar seus testes.

Antes de executar a simulação a entrada “INSTRUÇÃO” deve ser substituída pela saída da memória de instruções, a qual é endereçada por PC.

Depois dos arquivos carregados nas memórias habilite a simulação, clique nos botões “Reset Program” e “Zero Register File” para limpá-los. E, em seguida, escolha “Tick Once” no clock para avançar cada instrução.

Quando você começa a simular, sua primeira instrução estará no processador. Certifique-se de que tudo está funcionando. Verifique as entradas e saídas, os MUXes, os bits de controle. E, finalmente, verifique se o valor está sendo escrito de volta no arquivo de registro e o registro sendo escrito. (Ambos são exibidos nas “pontas de prova” enquanto o processador é executado.)

Quando seu programa ficar sem instruções, você pode verificar novamente o conteúdo do banco de registradores abrindo-o.

Dicas para debugar:

- Não avance além da primeira instrução até que esteja funcionando corretamente.
- Verifique se os valores de todos os registradores (registrador selecionado e seu respectivo valor) estão corretos e se a operação da ULA está correta.
- Use a ferramenta dedo (mãozinha) para ver os valores nos fios do processador.
- Trace os valores de volta ao primeiro lugar onde eles estão errados, e então vá para essa parte para ver o que está acontecendo.
- Você pode visualizar o conteúdo do banco de registradores visualizando dentro dele também.
- Escreva o que você espera que aconteça e verifique. Não adivinhe.
- Quando você precisar reiniciar o programa, clique em Restart Program e clique em Zero Reg File.
- Se você reiniciar a simulação, a memória será zerada e você terá que recarregar os dados da memória.

REFERÊNCIAS:

HENNESSY, John. Organização e Projeto de Computadores. 5 ed. Rio de Janeiro: Elsevier. Grupo GEN, 2017. E-book. ISBN 9788595152908. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788595152908/.

FOYER, Per. Notas de aula de “Computer Architecture - 1DT016”. Disponível em: https://xyx.se/1DT016/labs.php#lab1