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

3.1 – Trabalho de Implementação

Trabalho em dupla ou individual
Data de entrega (via SIGAA): 02 de Outubro de 2023
Pontuação: 5,0 pontos na III Unidade

IMPORTANTE:

- **O trabalho deve ser apresentado em sala de aula para o professor para ter a nota contabilizada**
- **O trabalho deve ser postado no SIGAA até o início da aula a qual apresentará o trabalho**

Descrição:
Construir uma versão *pipeline* do processador MIPS básico construído na unidade anterior, usando o Logisim. Tal processador é uma versão simplificada que suporta apenas as instruções **LW, SW, BEQ, ADD, SUB, AND, OR** e **SLT**, com pipeline.

Para isso são necessárias 4 coisas:

1) Ter uma versão do MIPS básico com as partes ligadas e funcionando adequadamente para as instruções listadas anteriormente
2) Identificar os diferentes estágios de execução de uma instrução MIPS
3) Adicionar registradores entre os estágios para armazenar temporariamente os dados e sinais de controle
4) Verificar que seu projeto funciona e executa vários programas

No projeto do LogiSim deve ter **os nomes dos componentes** e o arquivo deve ser salvo também com o nome dos componentes. Exemplo: NomeA-NomeB-mips-datapth-lab.circ.
Este arquivo deve ser submetido na tarefa do SIGAA dentro do prazo.

Para começar:

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

Instale o Logisim, versão 2.7.1, no seu computador, também disponível no SIGAA. Para executar este software é necessário que em seu computador também esteja instalado o *Java Runtime*.

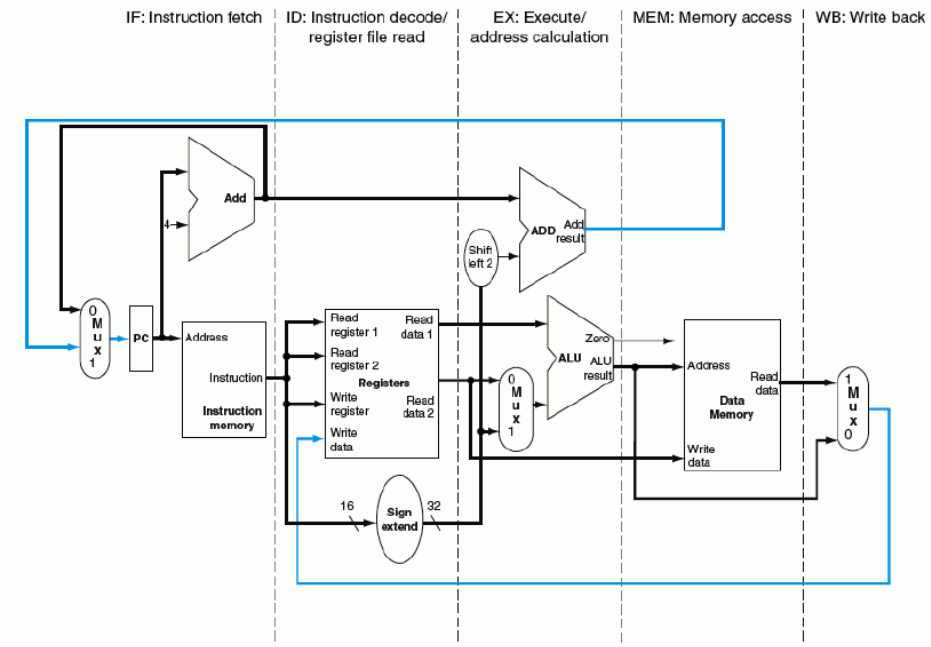
Utilize como ponto de partida o arquivo do projeto anterior (Unidade II), caso contrário utilize a versão no arquivo mips-datapath-lab.zip e antes de desenvolver o pipeline monte toda a estrutura de execução ciclo único. No arquivo .zip será encontrado os demais arquivos:

- mips-pipeline-lab.circ: um “esqueleto” do processador MIPS básico com importantes partes faltando propositalmente.

- 1234.mem: um arquivo de memória hexadecimal que contém os valores 1, 2, 3, 4 nas primeiras 4 palavras da memória
- test-code.mem: um arquivo que contém um programa de teste simples para verificar seu projeto

Passo 1: adicionar os registradores de pipeline:

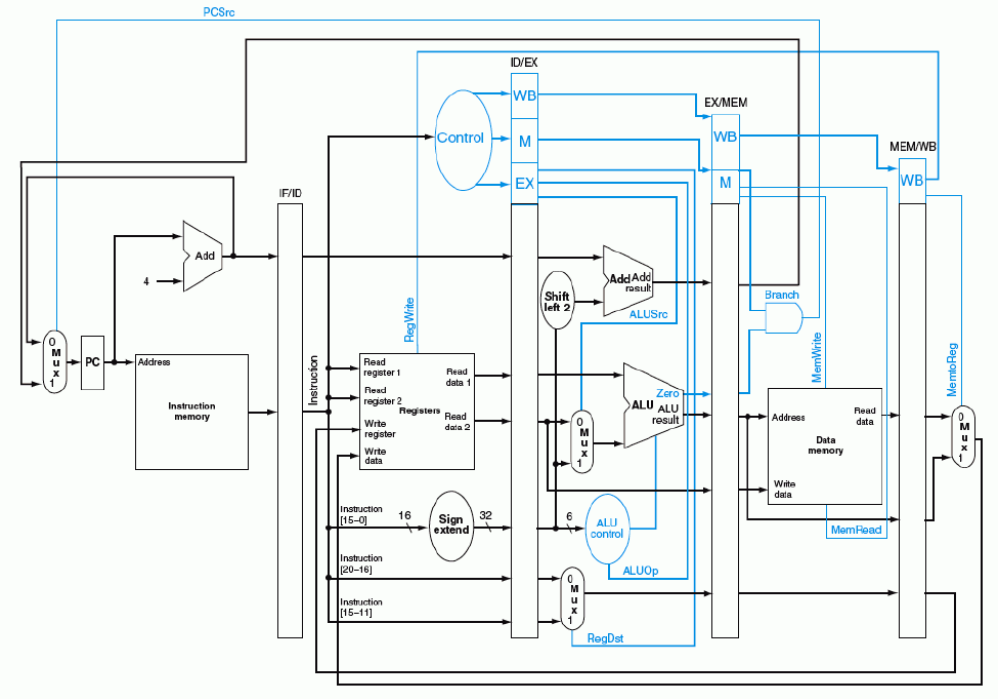
Há 5 estágios de pipeline no MIPS. A Figura a seguir indica onde os registradores de pipeline devem ser inseridos na versão básica de ciclo único.



Tais estágios serão referidos como **IF** (Instruction Fetch), **ID** (Instruction Decode), **EX** (Execute), **MEM** (Data Memory acess) e **WB** (Write Back). E os registradores com os nomes dos 2 estágios onde ele se encontra, fazendo a comunicação dos dois estágios. Assim os registradores de pipeline (RP) terão os nomes: **IF/ID**, **ID/EX**, **EX/MEM** e **MEM/WB.**

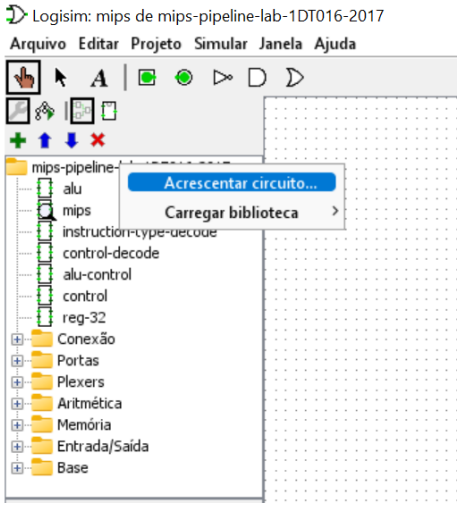
Para fazer esses estágios de pipeline funcionarem, é necessário colocar *registradores de pipeline* (RP) entre eles. Os RP armazenam temporariamente a instrução e qualquer valor necessário para o estágio, os quais vão sendo movidos através do pipeline a cada ciclo. (Ex: se os valores lidos do banco de registradores ou a instrução é decodificada, será necessário armazenar essas informações nos RP e em cada estágio tem acesso a elas). Os valores e sinais de controle são movidos de um estágio para o próximo a cada ciclo de *clock*. Sua primeira tarefa é adicionar esses RP e saber o que armazenar neles em cada estágio (os fios que cruzam os estágios dão uma boa dica). Tenha certeza de entender quais dados precisa armazenar e onde armazená-los na figura acima antes de implementar.

Sua arquitetura deve ficar semelhante a Figura a seguir.



As próximas figura mostram uma visão alto-nível do processador com os registradores adicionados. As caixas retangulares situada entre os estágios incluem todos os registradores e dados do caminho de dados, enquanto as linhas azuis indicam os sinais de controle, organizados pelo estágio em execução. Na figura anterior entre um estágio e outro do pipeline ter apenas 1 registrador, na prática eles serão implementados por vários, uma vez que no LogiSim o tamanho máximo dos registradores é de 32 bits. Para isso vamos implementar circuitos com entradas e saídas bem definidas, as quais internamente possuam a quantidade de registradores e quantidades de bits necessário para cada registrador de pipeline.

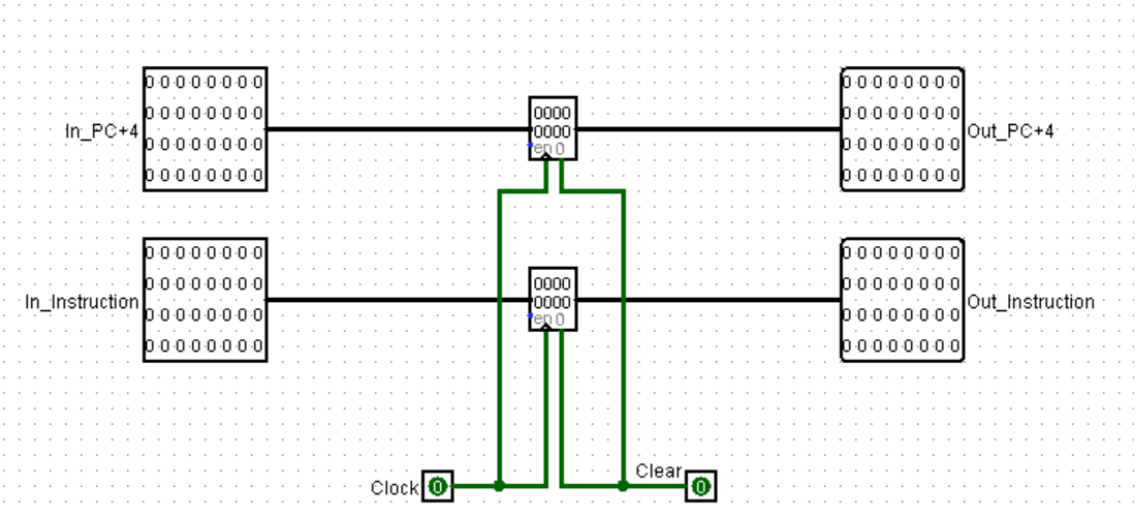
Para criar um circuito no Logisim, clique com o botão direito do mouse sobre o título do seu projeto e escolha a opção “Acrescentar circuito...”, conforme a figura



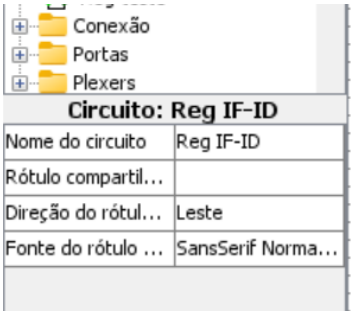
Em seguida defina os pinos de entrada e os de saída (com a quantidade de bits de dados adequados), ligue os componentes necessários que façam a computação dos valores recebidos nas entradas e produzam as saídas. Vamos definir o primeiro registrador de pipeline (IF/ID) como exemplo.

IF/ID

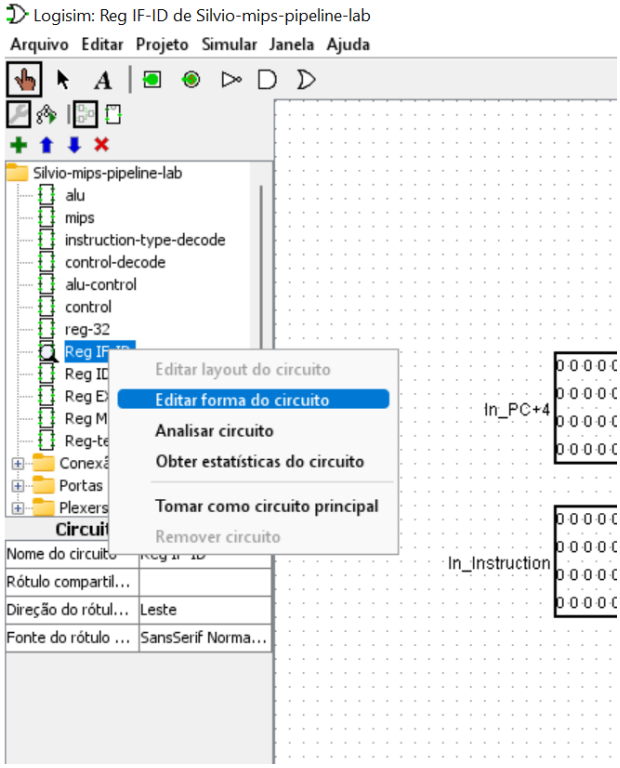
Vamos começar com um bem fácil. Para este serão necessários apenas 2 registradores: um para o valor “PC+4” e um para a instrução lida da memória. No arquivo “mips-pipeline-lab.circ” tem um exemplo de um deles. Nesse caso, ambos os registradores são de 32 bits de largura, e não há nenhum sinal de controle para ser salvo. Você precisará adicionar o segundo registrador alinhado próximo a saída da memória de instrução no seu projeto. Logo, criamos um circuito chamado “Reg IF-ID” e o definimos conforme a figura:



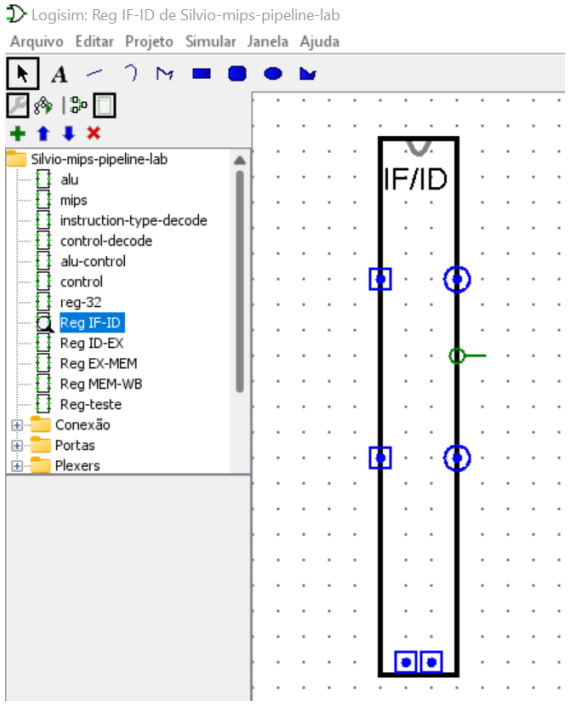
Então, temos 2 entradas de dados de 32 bits (In_PC+4 e In_Instruction) e 2 entradas de 1 bit (clock e clear). E temos 2 saídas (Out_PC+4 e Out_Instruction), ambas de 32 bits. Entre as entradas e saídas de 32 bits estão 2 registradores de 32 bits, responsáveis por armazenar uma nova informação a cada *clock*. Nas propriedades do circuito dê o nome “Reg IF-ID”, conforme o exemplo:



Após a definição do circuito, podemos definir seu formato, o qual facilitará as conexões na arquitetura principal. Para isso, clique com o botão direito do mouse sobre o nome do componente conforme a figura:



Em seguida escolha um formato retangular e posicione as entradas e saídas de modo que as entradas fiquem do lado esquerdo da figura e as saídas do lado direito, conforme a figura. Inclua também um texto no topo do circuito o nome dele “IF/ID” para aparecer na arquitetura geral.



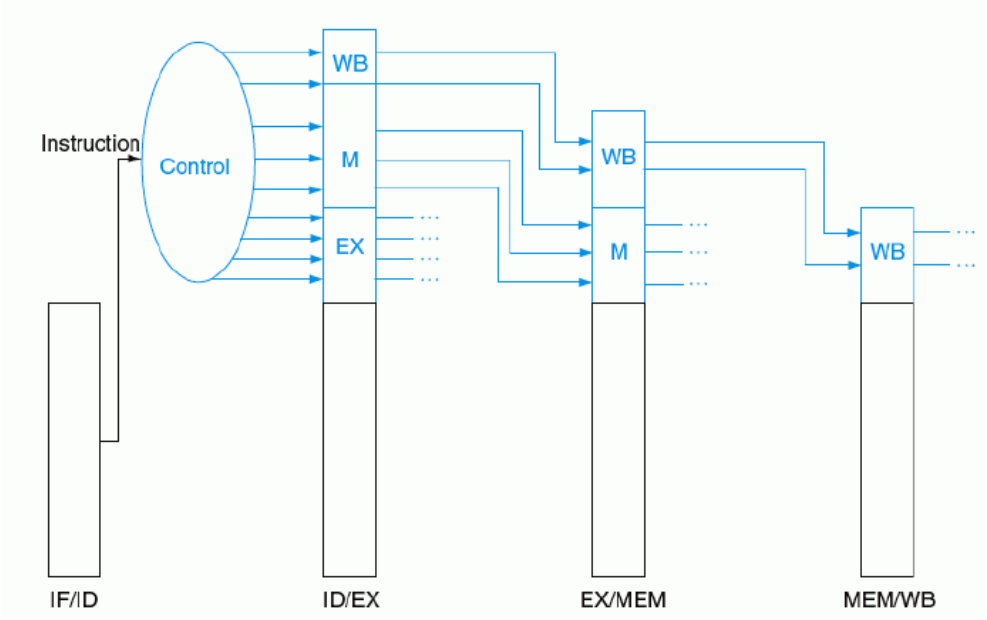
Siga o mesmo procedimento para os demais registradores de pipeline.

ID/EX

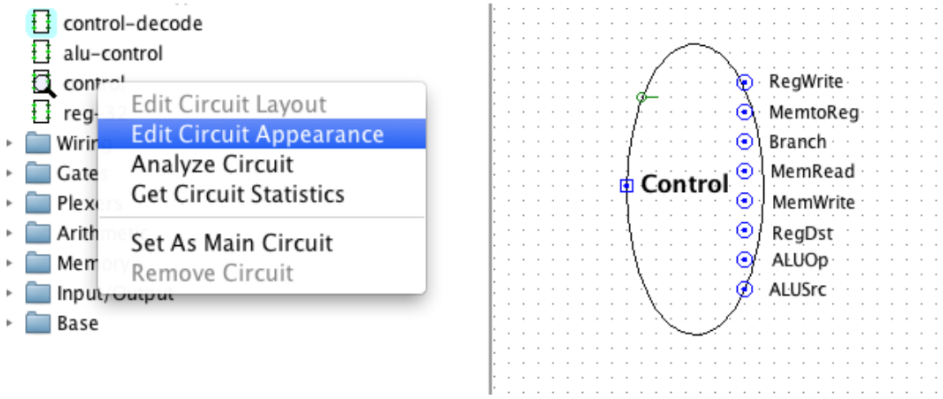
Aqui é onde a coisa começar a ficar um pouco bagunçada. Nós precisamos dar atenção especial para os dados/controle para entender o que está acontecendo, mas não se preocupe,

se você fizer da forma correta, o resto será muito simples! Pense cuidadosamente sobre quais dados precisam ser gerados no estágio ID e enviar ao longo dos próximos estágios antes de começar a construir. Veja o *dataflow* e identifique os dados e sinais de controle que precisam ser armazenados nesse estágio.

- Em relação aos dados, nós precisamos salvar o próximo valor de PC (vindo do estágio IF), os valores lidos do banco de registradores, bem como a saída do módulo de extensão de sinal. Esse procedimento é muito similar a parte anterior, apenas adicionar alguns registrados alinhados às saídas. Entretanto, nós também precisamos salvar alguns bits da instrução original em diferentes registradores, tais como os bits [20-16] e [15-11]. Ao adicionar os registradores tenha cuidado com os ajustes das larguras, uma vez que você precisa de 32 bits para uns e para outros não.
- Aqui é o primeiro estágio que precisa salvar sinais de controle para usar nos estágios seguintes. Se você olhar de perto, no livro, verá uma figura como a apresentada a seguir.



Os sinais de controle são separados por quais estágio serão usados. A unidade de controle é a mesma fornecida na implementação anterior (ciclo único), mas cuidado! A ordem desses sinais não corresponde a figura do controle pipeline do livro. Isso significa que os 2 sinais da unidade de controle que você implementou anteriormente **não são necessariamente** os mesmos 2 sinais de controle descritos na figura aqui apresentada. Seu primeiro trabalho é mudar o circuito do controle para colocar os sinais de saída na ordem na qual usará na versão pipeline. Uma forma de fazer isso é clicar com o botão direito do mouse na unidade de controle e selecionar “Editar forma do circuito”, como mostrado na figura a seguir. Isso permitirá você reordenar os pinos de saída da unidade de controle como você precisa. A segunda alternativa é reordenar os registradores que armazenam os sinais de controle, mas não recomendamos isso, uma vez que é criado uma grande quantidade de emaranhamento de fios.

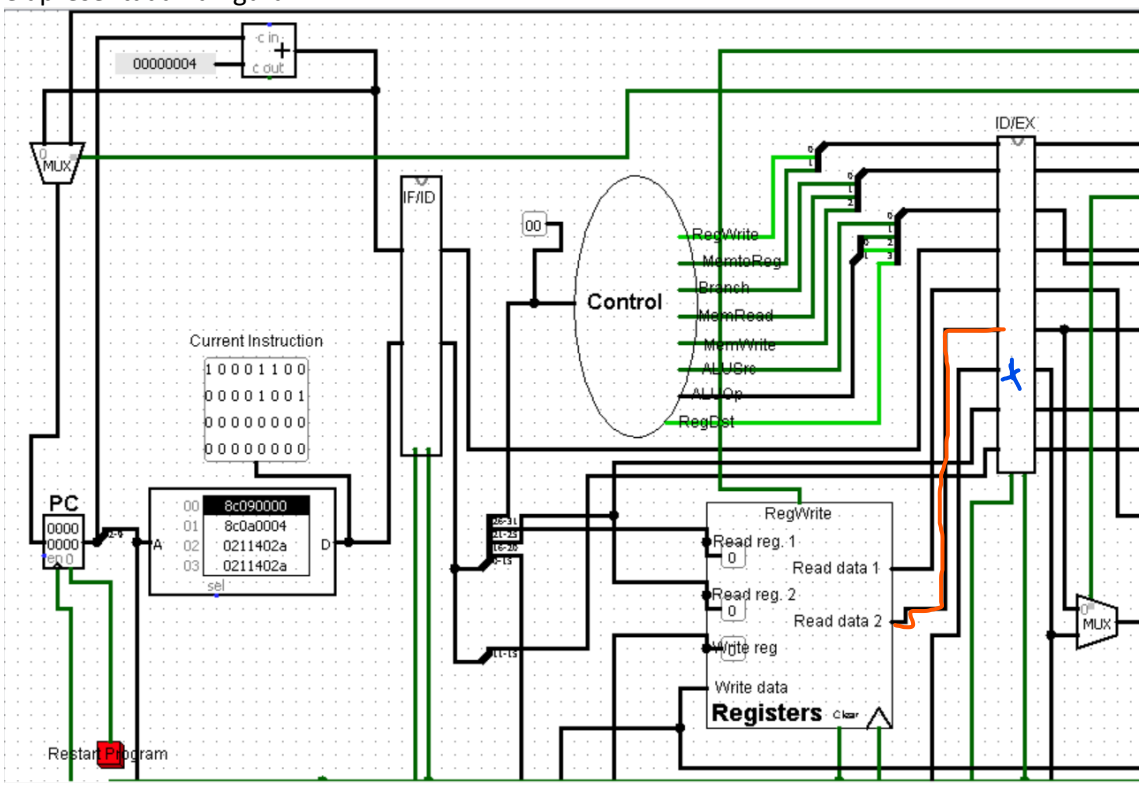


Depois de reorganizar sua unidade de controle, você precisa adicionar o RP para os sinais.

EX/MEM e MEM/WB

Agora que você já ganhou experiência adicionando os registradores anteriores, pode apenas repetir o procedimento para os próximos 2 estágios do pipeline. Lembre de identificar quais sinais de controle e dados são necessários em cada e estágio e a largura dos registradores apropriadamente. Perceba que os sinais de controle não estão sempre no topo, acima dos registradores de dados, como no caso do sinal *zero*.

Após criar cada registrador de pipeline, ligue-os na arquitetura geral do MIPS, armazenando as informações geradas em cada estágio e que será consumida pelo próximo estágio, a cada ciclo. Um exemplo de parte da arquitetura com 2 desses registadores ligados para formar o pipeline é apresentada na figura:



Passo 2: testar com código

Para testar seu processador, fornecemos 3 códigos de programas (test-code-1.mem, test-code-2.mem e test-code-3.mem):

***Test code 1*:**
lw $t1, 0($zero)
sw $t2, 0($zero)

***Test code 2*:**
lw $t1, 0($zero)
sw $t1, 12($zero)

***Test code 3*:**
lw $t1, 0($zero)
lw $t2, 4($zero)
add $t3, $t1, $t2
add $t3, $t1, $t3
sw $t3, 12($zero)

Você pode usar o *assembler* online http://alanhogan.com/asu/assembler.php ou o MARS para gerar o código montado e salvar seus arquivos .mem para testar no LogiSim.

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 entrar na unidade de controle enquanto seu processador está simulando para ver onde estão os erros.
- 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