ALU e Regfile

Visão Geral

Nesse projeto você irá utilizar o Logisim para implementar duas unidades básicas de um processador simples: ALU e Register File. Este processador utiliza um subconjunto das instruções de MIPS chamado RPIS (Reduced Processor Instruction Set). Esta ISA utiliza 8 registradores com 16 bits de dados e um espaço de endereçamento de memória também de 16 bits.

O datapath deste conjunto de instruções é bastante semelhante ao datapah MIPS que você aprendeu em aula, com algumas modificações para acomodar nosso número reduzido de registradores e espaço de endereços.

RPIS (Reduced Processor Instruction Set)

1) Register File (Regfile)

Sua tarefa é implementar os 8 registradores da arquitetura RPIS (utilize os registradores disponíveis no Logisim).

- Seu regfile deve ser capaz de gravar ou ler de cada registrador especificado em uma determinada instrução, sem afetar os outros registradores. Existe uma exceção trivial: seu regfile não deverá escrever sobre `$0`, mesmo que uma instrução tente.
- O Registrador Zero sempre deve ter valor `0x0`.
- O clock sempre deve ser conectado diretamente à entrada de clock dos registradores, sem passar por qualquer combinação lógica (i.e., não deve passar por qualquer porta lógica e/ou bloco do Logisim).

Os registradores e seus respectivos números estão listados abaixo. Seus propósitos são os mesmos dos de MIPS.

Registrador	Nome
0	$0
1	$ra
2	$s0
3	$s1
4	$s2
5	$a0
6	$v0
7	$sp

Foi lhe fornecido um esqueleto do regfile no arquivo `regfile.circ`. O circuito tem seis entradas:

Nome da entrada	Bits	Descrição
Clock	1	Fornece o clock. Este sinal deve ser ligado diretamente à entrada de clock dos registradores e pode ser enviado à subcircuitos desde que não passe por qualquer porta lógica (i.e., não pode ser invertido, fazer um AND com outro sinal ou qualquer coisa do tipo).

Nome da entrada	Bits	Descrição
Write Enable	1	Determina se algum dado deve ser escrito na próxima subida de clock (i.e., quando `Clock` for 1).
Read Register 1	3	Determina qual registrador enviar seu valor para a saída `Read Data 1` (veja abaixo).
Read Register 2	3	Determina qual registrador enviar seu valor para a saída `Read Data 2` (veja abaixo).
Write Register	3	Determina o registrador que deve receber o valor de `Write Data` na próxima subida de clock, assumindo que `Write Enable` é 1.

O regfile também possui 6 saídas:

Nome da saída	Bits	Descrição
Read Data 1	16	Destinado ao valor do registrador especificado em `Read Register 1`.
Read Data 2	16	Destinado ao valor do registrador especificado em `Read Register 2`.
$s0 Value	16	Sempre destinado ao valor do registrador $s0.
$s1 Value	16	Sempre destinado ao valor do registrador $s1.
$ra Value	16	Sempre destinado ao valor do registrador $ra.
$sp Value	16	Sempre destinado ao valor do registrador $sp.

OBS.:

- As entradas e saídas servem como Debug/Teste e serão utilizadas no algoritmo de correção, por isso, não às mova de lugar nem às renomeie.
- As saídas `$s0`-`$s2`, `$ra` e `$sp` não estão presentes em uma implementação real de um regfile. Em nosso caso, elas servem como Debug/Teste como dito acima.
- Seu `regfile.circ` também deve caber no circuito `regfile-harness.circ`. Isso significa, novamente, que você não deve alterar as entradas e saídas, nem suas posições.
- Caso precise de mais espaço, poderá utilizar os túneis do Logisim ou desenvolver a parte interna do Regfile em outro arquivo `.circ` e importá-lo em `regfile.circ`, conectando suas respectivas entradas e saídas (não se esqueça da especificação do clock).

DICAS:

- Muxes e demuxes são seus amigos.
- Devo avisar-lhe que a entrada `enable` de seus muxes não deve ser utilizada (i.e. não deve estar conectada a qualquer sinal/fio).
- Devo avisar, também, que você deve desativar a opção `three-state` em todas as entradas, saídas, registradores e afins.

2) Arithmetic Logic Unit (ALU)

Sua segunda tarefa é criar uma ALU que suporte todas as operações necessárias às instruções em nossa ISA (que será melhor descrita no próximo trabalho). A maior parte da nossa ISA consiste em instruções MIPS (podem haver novas instruções e formatos adicionais). Felizmente, você não precisa descobrir todas as operações necessárias, pois nós fornecemos uma lista delas abaixo. Observe que nós ignoramos o Overflow, assim como o MIPS faz com as instruções unsigned (sem sinal).

Também lhe fornecemos um esqueleto de uma ALU em `alu.circ`. O circuito tem três entradas:

Nome da entrada	Bits	Descrição
X	16	Valor de X para as operações da ALU.
Y	16	Valor de Y para as operações da ALU.
Switch	3	Seleciona qual operação a ALU deve realizar (veja abaixo a lista de instruções e seus respectivos valores de switch).

E 3 saídas, já que estamos ignorando o Overflow:

Nome da saída	Bits	Descrição
Equal	1	1 se `X` e `Y` são iguais, 0 caso contrário.
Result	16	Resultado da operação na ALU.
Result2	16	Contém os 16 bits superiores do resultado de uma multiplicação quando a função é multiplicação. Contém 16 bits de qualquer valor em outras operações.

Como mencionado anteriormente, segue abaixo a lista de operações que você precisa para implementar (de acordo com o `Switch`) a ALU. É aconselhável que você utilize os blocos disponíveis no Logisim para implementar as operações de aritmética.

Switch	Instrução
0	sll: `Result` = `X` << `Y`
1	srl: `Result` = `X` >>> `Y`
2	add: `Result` = `X` + `Y`
3	and: `Result` = `X` & `Y`
4	or: `Result` = `X` | `Y`
5	xor: `Result` = `X^Y`
6	slt: `Result` = (`X` < `Y`) ? 1 : 0 Signed (i.e., 1 Signed 16 bits = 1111 1111 1111 1111)
7	mult: `Result` = `X*Y`[15:0]; `Result2` = `X*Y`[31:16]

OBS.:

- Como dito anteriormente, se não é necessário `Result2` no resultado de uma operação, então ele pode assumir qualquer valor.
- Para sll e srl, os circuitos de deslocamento no Logisim só mudarão para o número de bits de dados, portanto, para os nossos registros de 16 bits, você só precisa se preocupar em gerenciar mudanças de até 15 bits. Isso implica em modificar a entrada `Y` antes de inseri-la no deslocador, de modo que apenas os quatro bits inferiores de `Y` sejam usados como o valor de mudança enquanto os bits de ordem superior devem ser ignorados.
- O circuito multiplicador incorporado no Logisim é signed (quando operado em números de 16 bits)! Não é esperado que você implemente seu multiplicador do zero.
  - Use o bloco de multiplicação disponível no Logisim e não se preocupe com isso.
- A saída `Equal`, deve sempre produzir o resultado de comparação correto independentemente do valor `Switch`.

- As entradas e saídas servem como Debug/Teste e serão utilizadas no algoritmo de correção, por isso, não às mova de lugar nem às renomeie.
- Seu `alu.circ` também deve caber no circuito `alu-harness.circ`. Isso significa, novamente, que você não deve alterar as entradas e saídas, nem suas posições.
- Caso precise de mais espaço, poderá utilizar os túneis do Logisim ou desenvolver a parte interna da ALU em outro arquivo `.circ` e importá-lo em `alu.circ`, conectando suas respectivas entradas e saídas.

Notas de Apoio

Observações sobre o logisim

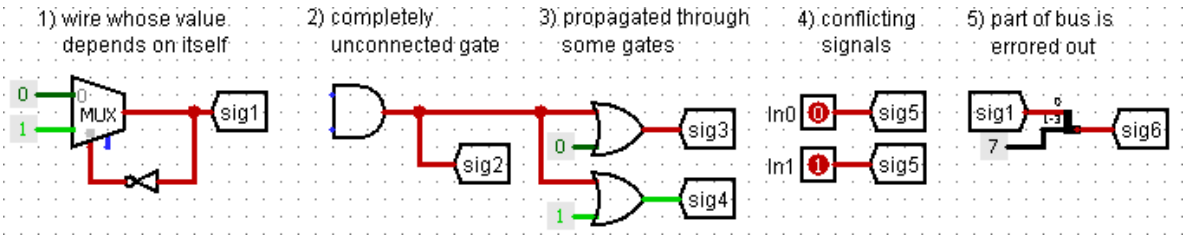
Se você estiver com bugs no Logisin, FECHE e ABRA o circuito novamente! Não perca seu tempo tentando reparar um problema que não é seu. No entando, caso reiniciar não tenha resolvido o problema, é provável que exista algum problema de implementação em seu circuito.

Caso o problema continuar após modificações, tendo certeza que não é um problema seu, entre em contato:

- Denilson Amorim: denimorim@gmail.com
- Thalles Medrado: thallesyann (at) hotmail (dot) com

Ainda sobre o logisim, vale a pena ressaltar os seguintes pontos:

- NÃO ligue o clock em portas lógicas! Esta é uma péssima prática de design na montagem de circuitos, não aconselhamos que faça algo assim e, por isso, iremos penalizar seu projeto.
- Seja CAUTELOSO ao copiar e colar circuitos entre janelas do LOGISIM. O logisim é conhecido por problemas com este tipo de tarefa.
- Quando você importa outro arquivo (Project -> Load Library -> Logisim Library ...), ele aparecerá como uma pasta no painel de visualização da esquerda. Os arquivos de esqueleto já devem ter importado os arquivos necessários.
- Alterar atributos antes de setar um componente altera as configurações padrões do componente. Então, se você for colocar muitos pinos de 32 bits, pode ser útil alterar sua configuração padrão. Se quiser alterar apenas esse componente específico, coloque o componente e depois altere sua configuração.
- Quando você altera as entradas e saídas de um subcircuito que você já colocou no main, o Logisim irá adicionar/remover automaticamente as portas quando retornar ao main e isso, por vezes, muda o próprio bloco. Se haviam fios conectados, o Logisim também fará o deslocamento automático destes, o que pode ser extremamente ruim em alguns casos. Antes de alterar as entradas e saídas de um bloco, às vezes pode ser melhor separar primeiro todos os fios dele no main.
- Os erros de sinal (fios vermelhos) são obviamente ruins, mas tendem a aparecer em tarefas de fiação complicadas, como as que você estará implementando aqui. É bom estar ciente das causas comuns durante a depuração:



- O Logisim oferece algumas funções para automatizar a implementação do circuito dada uma tabela de verdade, ou vice-versa. Embora permitido, o uso desse recurso é desencorajado. Lembre-se de que você não terá um laptop executando Logisim na prova final.

Testando

Para testar seu projeto basta rodar:

```
make alu-regfile
```

Este script irá realizar os testes básicos do projeto e lhe oferecer os resultados (em inglês).

É aconselhável que realize mais testes manualmente para corner cases.

Submissão

Você deve submeter apenas os circuitos necessários para o funcionamento correto de sua `alu.circ` e `regfile.circ` (inclusive) dentro de um zip com o nome da equipe.

Ex.: Suponha que você modificou apenas os circuitos `alu.circ` e `regfile.circ`, sua submissão `equipe.zip` deve conter apenas estes dois circuitos.

OBS.: Note que todos os arquivos necessários devem estar na raiz do zip, a submissão que não estiver de acordo estará sujeita a punição.

Avaliação

O processador será avaliado em maior parte por um sistema automático. Se algum dos seus testes falharem, tentaremos ver se é um problema simples, e então consertar. Nós então daremos a nota automática, menos um percentual pela gravidade do problema. Por esse motivo, tente deixar o circuito o mais legível e funcional possível.

Poderá haver penalidade por:

- Circuito em desacordo com especificações do projeto;
- Plágio; Ambas equipes serão penalizadas!
- Submissão fora do formato especificado.

Lembre-se de usar o **Logisim Evolution**! Caso contrário, seu circuito não será compatível com o sistema de testes.