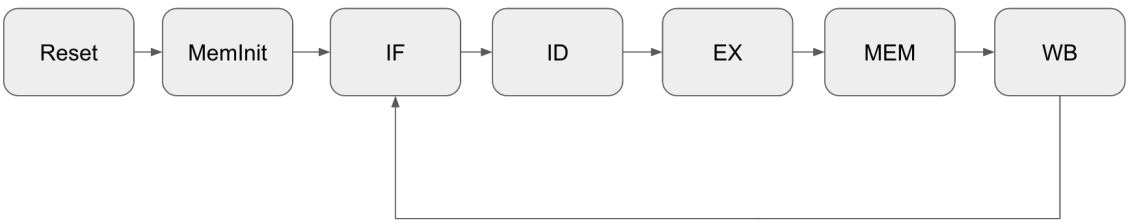
Trabalho Prático
2ª Entrega
Organização de Computadores II
2016/1

1 Objetivo

Complementar o processador MIPS multiciclo para funcionamento com uma gama maior de instruções e de forma correta.

2 O processador MIPS Multiciclo

Um processador MIPS pode ser resumido em uma máquina de estados onde as transições são controladas por um clock. As transições são realizadas de forma regulada para garantir que todas as estruturas tenham realizado suas operações antes de que novas operações sejam realizadas. Estruturas (como a memória) podem exigir um determinado número de transições para que os valores de saída sejam recebidos para determinadas entradas.



Cada estado tem uma função pré-definida:

- **Reset**: Estado alcançado apenas na inicialização ou reinicialização do processador. Apaga/Zera todo o banco de registradores e memória além de fazer que a máquina volte a primeira instrução.
- **MemInit**: Aguardando que a instrução inicial seja recuperada da memória de instruções, já que a resposta desta estrutura não é imediata.
- **IF - *Instruction Fetch***: Recupera uma instrução da memória de instruções e realiza o pedido para que a memória já carregue a próxima instrução (já que, como dito, a memória não responde imediatamente).
- **ID - *Instruction Decode***: Decodifica a instrução (determina o seu tipo) e o caminho de dados a ser utilizado.

- **EX - *Execute***: Executa a computação necessária.
- **MEM - *Memory***: Realiza as operações da memória de dados (se necessário).
- **WB - *Write Back***: Grava os resultados no registrador correspondente (do banco de registradores ou apontador de instrução - *PC*)

Neste primeiro momento não é necessário que se faça tratamento de hazards visto que somente uma instrução estará no caminho de dados (sendo processada pela máquina de dados). Todas as instruções deverão passar por todos os estados, mesmo que não realizem nenhuma operação nele. Isso facilitará a implementação da próxima entrega (Que prevê a transformação em um pipeline).

3 Processador base

Para esta etapa, será fornecido um código que descreve um processador base que já realiza algumas operações (especificamente, uma instrução de cada tipo). Você e seu grupo devem completar este com o código já entregue para a primeira entrega além de complementa-lo a fim de que ele seja coerente e completo.

4 Modificações/Adições necessárias

O processador dado como base deve ser alterado para condizer com as especificações corretas do funcionamento como um MIPS multiciclo.

4.1 Banco de registradores

O registrador de endereço 0 (00000 em binário) deve ser para leitura apenas (qualquer tentativa de escrita nele não deverá ser possível.

4.2 Memória de dados

As estruturas necessárias para a memória de dados devem ser descritas de forma similar à memória de instruções. O conteúdo de ambas pode ser (e normalmente será) diferente.

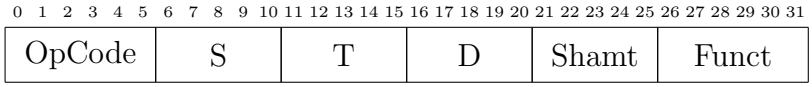
5 Instruções

O processador, ao fim desta etapa, deverá ser capaz (não necessariamente se limitar) de executar operações das seguintes instruções do padrão MIPS:

5.1 Tipos de Instruções

As instruções MIPS sempre tem 32 bits de comprimento e estão em 3 formatos bem definidos descritos a seguir.

5.1.1 Tipo R

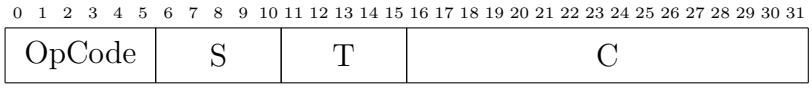


São, normalmente, instruções de operações com os valores de registradores. Sendo S e/ou T os endereço dos registradores dos operandos e D os de destino.

O processador resultado deve ser capaz de realizar as seguintes instruções do tipo R:

Instrução	Sintaxe	Descrição	OpCode	Func
Adição	add $d,$s,$t	$d = $s + $t	000000	100000
Subtração	sub $d,$s,$t	$d = $s - $t	000000	100010
E	and $d,$s,$t	$d = $s & $t	000000	100100
Ou	or $d,$s,$t	$d = $s | $t	000000	100101
Ou Exclusivo	xor $d,$s,$t	$d = $s $\wedge$ $t	000000	100110
Não Ou	nor $d,$s,$t	$d = $\sim$($s | $t)	000000	100111
Shift Esquerda Lógico	sll $d,$t,shamt	$d = $t << shamt	000000	000000
Shift Direita Lógico	srl $d,$t,shamt	$d = $t >> shamt	000000	000010
Salto Incondicional por registrador	jr $s	PC = $s	000000	001000

5.1.2 Tipo I

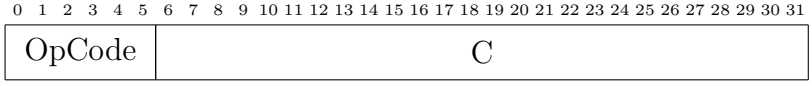


São instruções de operações imediatas (que dependem de valores constantes). S é normalmente o endereço do registrador base e T, o alvo. C é o valor constante. É importante processar o valor de C como complemento de 2 para ideal implementação de possíveis valores negativos.

O processador resultado deve ser capaz de realizar as seguintes instruções do tipo I:

Instrução	Sintaxe	Descrição	OpCode
Adição imediata	addi $t,$s,C	$t = $s + C	001000
Carrega Palavra	lw $t,C($s)	$t = MemDados[$s + C]	100001
Armazena Palavra	sw $t,C($s)	MemDados[$s + C] = $t	101011
Salto Condicional em igualdade	be $s,$t,C	if ($s == $t) PC = (PC+4)+(4*C)	000100
Salto Condicional em não igualdade	bne $s,$t,C	if ($s != $t) PC = (PC+4)+(4*C)	000101

5.1.3 Tipo J



São instruções de salto.

O processador resultado deve ser capaz de realizar as seguintes instruções do tipo J:

Instrução	Sintaxe	Descrição	OpCode
Salto incondicional	j C	PC = 4*C	000010
Salto incondicional com registro	jal C	$31 = PC + 4; PC = 4*C	000011

6 Considerações

A memória **deve** ser implementada de forma a utilizar a RAM da FPGA e deve ser enderecável por byte e não por palavra, da mesma forma que o código base. Utilize o *MegaWizard* do Quartus para criar o bloco de memória e seu respectivo inicializador.

O código exemplo demonstra os valores dos registradores nos LEDs e displays de 7 segmentos da FPGA fornecida. Mantenha essa parte do código.

Devem ser elaborados conteúdos para a memória de instrução e de dados a fim de comprovar a execução de cada uma das instruções.

Ao se completar o código fornecido com o já entregue na primeira etapa, seu processador já deverá ser capaz de executar todas as instruções do tipo R que fazem uso da ULA. Também já estão implementadas uma instrução de cada um dos outros dois tipos.

7 Entrega

Data de entrega: 08/05/2016

Devem ser entregues o projeto do Quartus e um pequeno relatório contendo decisões, explicações feitas e testes feitos. É esperado que os testes tenham sido efetuados na FPGA fornecida.