CI210 - Projetos Digitais e Microprocessadores
Trabalho Prático

Profa. Michele Nogueira - michele@inf.ufpr.br
Monitor: Leonardo Melniski - leomelniski@gmail.com

[1]Departamento de Informática - NR2/UFPR

Data de Entrega: dia 03 de Dezembro de 2013 (todos devem entregar nesta data). (Não serão aceitos trabalhos fora do prazo!)

Aulas reservadas para apresentação do trabalho prático são as dos dias: 03 e 05 de Dezembro de 2013 no Lab. Um sorteio será realizado para definição da ordem de apresentação

Lembrem: plágios de trabalhos não serão tolerados.

Apresentação

Utilizando como base o picoProcessor (veja código fonte no sítio web da disciplina, seção 'Material de Referência'), você deverá implementar, dentre as opções de projetos abaixo, pelo menos **duas**. Este trabalho prático deverá ser feito em duplas e exclusivamente em VHDL.

Dica: Antes de tentar implementar alguma nova instrução, procure entender o funcionamento completo do picoProcessor e identificar as fases de *Busca de instrução*, *Decodificação de instrução*, *Leitura de memória (quando aplicável)* e *Execução da instrução*. É importante também **projetar** antes de **implementar** :-).

Na seção 'Material de Referência' do sítio Web da disciplina, você encontrará a documentação do picoProcessor.

Modificações Propostas:

1. Shifter

Implementar um shifter completo, de 0-31 bits, em qualquer direção, de forma que seja executado em apenas um ciclo. Ajustar o decodificador de instrução para funcionar corretamente com as novas instruções (sll, sllv, srl, srlv, sra e srav)

2. Carry look-ahead adder

Projetar e implementar um look-ahead carry adder (Veja: CLA) de forma que diminua o atraso de propagação do carry de 64-gates para algo próximo a 7-gates, fazendo com que as operações de soma/subtração/comparação demorem tanto quanto as operações lógicas. Ajustar o Controle para atingir esse tempo.

3. Máquina de Estados Finitos - Multiciclo

Reprojetar a sequência de instruções para que utilizem uma máquina de estados finitos (*FSM*), de forma que as instruções mais lentas não atrasem as mais rápidas. Verificar se todas as demais instruções continuam sendo executadas corretamente.

4. Multiplicação

O picoProcessor não possui o hardware necessário para realizar a multiplicação. Implementar um instrução de multiplicação (mult/multu), que inicia quando a instrução é emitida e roda assincronamente até o término da mesma, guardando o resultado do produto nos registradores Hi e Lo.

Note que o resultado da multiplicação de dois números de 8bits é um número de 16bits, daí a necessidade de utilizar os registradores Hi e Lo.

5. Divisão

O picoProcessor não possui o hardware necessário para realizar a divisão. Implementar uma instrução de divisão (div/divu) que inicia quando a instrução é emitida e roda assincronamente até o termino da mesma, guardando o coeficiente no registrador Hi e o resto no registrador Lo.

6. Soma em ponto flutuante

Implementar as operações de soma e subtração em ponto flutuante (*add.s* e *sub.s*) e as duas conversões (*cvt.s.w* e *cvt.w.s* que usam muito da mesma lógica);

7. Multiplicação em ponto flutuante

Implementar a instrução para multiplicação em ponto flutuante (*mul.s*)

8. Byte e Half-word Load/Store

O picoProcessor atualmente somente faz *load* e *store* para palavras inteiras. Implementar a lógica extra para realização de operações de memória com palavras parciais.

OBS: Em breve serão divulgadas as orientações para entrega do trabalho e os critérios a serem utilizados na avaliação dos mesmos.