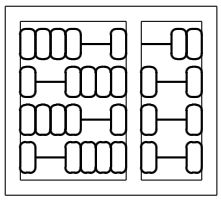

MC542
Organização de Computadores: Teoria e Prática

Trabalho 1

IC – UNICAMP

1 Entrega

Este trabalho deverá ser realizado em **grupo de até seis alunos**. Um membro do grupo (**e somente um**) deverá mandar um e-mail, até 04/11/2011, para ducatte@ic.unicamp.br com *subject*: mc542: MIPS-Pipeline - Grupo e no corpo da mensagem os RAs dos componentes do grupo (um RA por linha).

A data de entrega do trabalho será anunciada oportunamente na página do curso.

Escreva o seu projeto, com nome **mips** e um *testbench* **tb-mips**, realize simulações e apresente os resultados em um relatório (máximo de 5 páginas) a ser entregue junto com os arquivos vhdl. Além do relatório compacte o diretório do seu projeto e envie por e-mail para ducatte@ic.unicamp.br com *subject*: mc542: mips.

2 Objetivo

O objetivo deste trabalho é projetar e simular um processador MIPS pipeline básicos usando VHDL. Neste projeto iremos projetar e simular uma versão do processador MIPS apresentada no livro texto, com mecanismo de detecção de *hazard* e *forwarding* e reduzindo a penalidade devido ao *hazard* de controle a um ciclo. O processador deverá ser capaz de executar no mínimo as seguintes instruções: **load, store, add, addi, addu, sub, subi, subu, and, or, xor, slt, sltu, j, jal, beq**. Use como *Opcode* e *Funct* os valores fornecidos no apêndice B do livro texto.

Para simplificar o projeto iremos supor que a memória de instruções e de dados é externa ao nosso sistema, assim você não precisa modela-la (tarefa que não é trivial). Também iremos assumir que instruções e dados são de 32 bits. Nesta versão simplificada, assumir para os passos de *instruction fetch* e *data fetch* que os valores a serem carregados no respectivo registrador de pipeline estão disponíveis nas entradas *Instruction* e *Data* ou seja estes valores serão providênciados diretamente na simulação sem ser preciso ler uma memória.

Utilize como entidade base a entidade dada abaixo.

```
Entity mips is
  generic(nbits : positive := 32);
  port(Instruction : in std_logic_vector(nbits -1 downto 0);
       Data        : in std_logic_vector(nbits -1 downto 0);
       clk         : in std_logic;
       reset       : in std_logic;
       PCF         : out std_logic_vector(nbits -1 downto 0);
       ALUOutM     : out std_logic_vector(nbits -1 downto 0));
       WriteDataM  : out std_logic_vector(nbits -1 downto 0);
       MemWriteM   : out std_logic);
End mips;
```

O projeto conterá um *Register File*; ULAs (principal e para cálculo do novo valor do PC; PC; unidade de controle (*ALU Decoder* e *Main Decoder*, registradores auxiliares, inclusive os de pipeline etc implementados como componentes. (Use, com as modificações necessárias os módulos projetados nos labs anteriores.)

OBS.:

1) Para testar seu *datapath* enquanto a instrução *load* não estiver implementada pode-se usar o sinal de **reset** para além de colocar a unidade de controle no **estado inicial** inicializar os (ou alguns) registadores do banco de registradores para realizar a simulação de instruções tipo R.

2) Durante o reset o *PC* deve receber o valor 0H ié o *datapath* começa a sua execução pelo endereço 0h.

3 Concorra a Pontos Extras

Implemente as memórias de instrução e de dados e/ou extensões ao projeto básico acima, com intuito de aproxima-lo ao *datapath* pipeline apresentado no livro texto (conjunto de instruções mais completo, leitura e escrita de memória etc. Caso seja necessário, devido às extensões implementadas, acrescente à entidade base os sinais de entrada e/ou saída necessários.

Desenvolva o seu projeto em um diretório cujo nome seja composto pelos RAs dos elementos do do grupo (separados por - e tenha dois sub diretï¿½rios (tb e vhdl) e os arquivos como mostrado abaixo:

```
xxxxxx              -- Seu RA
  xxxxxx.pdf        -- Relatório (de 1 a 3 páginas)
  tb
    compila_tb.sh   -- comandos para analisar e elaborar o testbench
    executa_tb.sh   -- comando para executar o testbench
    tb_controle.vhd -- arquivo vhdl principal do testbench
```

```
vhdl
  compila.sh      -- comandos para analisar e elaborar o projeto
  mips.vhd        -- arquivo vhdl principal com a descrição do mips
  .....vhd        -- arquivos vhdl com a descriçãoo dos componentes usados
  ...

  pkg_mips.vhd    -- arquivo vhdl que define o package associado ao projeto
                     (se foi usado)
```