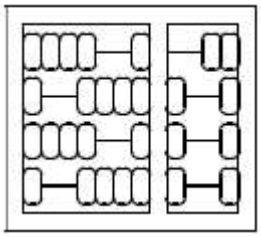

MC542
Organização de Computadores: Teoria e Prática
IC-UNICAMP

Projeto final

1 – Entrega

Este trabalho deverá ser realizado em grupo de dois alunos.

A data de entrega do trabalho é 23/nov/2005.

Faça o relatório contendo a descrição do projeto, comentários, código e resultados de simulação. Realize simulações comportamental e com *timing*.

2 – Objetivo

O objetivo deste trabalho é projetar e simular alguns componentes básicos do processador MIPS usando VHDL. Neste projeto iremos projetar e simular uma versão simplificada do processador MIPS multi-ciclos descrito no capítulo 5 do livro texto de *Patterson* e *Hennessy*.

O projeto será desenvolvido usando-se a ferramenta *Quartus* da *Altera* instalada nos laboratórios 1 e 2 do IC-3. Alternativamente você pode fazer *download* e instalar o *Quartus* em uma máquina pessoal (para acesso à página do Programa Educacional da Altera consulte a página do curso).

3 – Descrição do projeto

Implemente uma versão simplificada do *datapath* do processador MIPS multi-ciclos capaz de executar instruções *R-type*, como mostrado na figura abaixo. O *datapath* a ser implementado é um subconjunto do *datapath* multi-ciclo completo desenvolvido na seção 5.4 e mostrado na figura 5.33 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. Não implementaremos, nesta versão simplificada, o passo de *instruction fetch* e assumir que os valores a serem carregados em *Instruction Register* (IR) ou em *Memory Data Register* (MDR) estão disponíveis como entrada (sinais de entrada na entidade). Também iremos ignorar a parte relacionada com o PC, ou seja, o calculo de PC + 4 para a busca de uma nova instrução. (observação: o MDR não será usado para instruções tipo R).

Utilize como entidade base a entidade dada abaixo.

```
Entity R_datapath is
  generic(nbits : positive := 32);
  port(data : in std_logic_vector(nbits -1 downto 0);
     clk       : in bit;
     reset     : in bit;
     Zero      : out bit;
     Overflow  : out bit;
     ALUOut    : out std_logic_vector(nbits -1 downto 0));
End R_datapath;
```

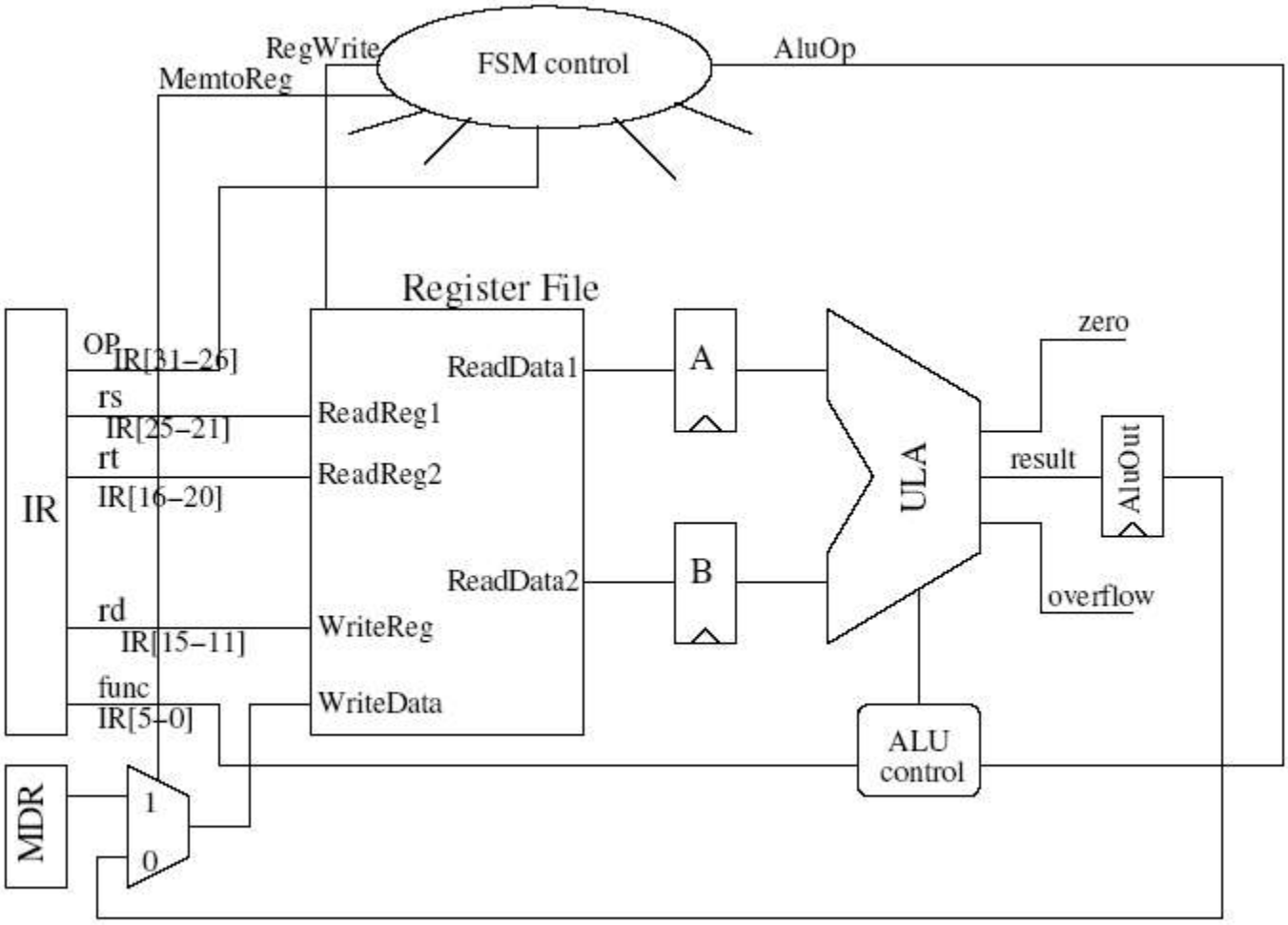

Figura 1: Datapth simplificado do MIPS para instruções R-type.

O projeto conterá um *Register File*, uma ULA, registradores auxiliares e as unidades de controle *ALU control* e *FSM control*. O *Register File* e a ULA são os componentes projetados nos exercícios anteriores e as unidades de controle são as unidades responsáveis por gerarem os sinais de controle necessários à execução das instruções *R-type*.

4 – Comportamento do Data Path

O *datapath* para instruções *R-type* deve ter o seguinte comportamento: cada fase é relacionada a um estado da *FSM control*, mostrada na figura 5.42 do livro texto. Como a nossa memória é externa o dado (instrução) estará disponível em *data* (as instruções serão editadas no arquivo de simulação) e deverão

ser carregadas em IR, assim devemos partir do estado 0 e também devemos retornar ao estado 0 durante o ciclo de execução de uma instrução. Para mais detalhes consulte o livro texto.

(estado 0) Neste ciclo deverá ser guardado em IR o valor disponível na entrada *data* da entidade.

(estado 1) Neste ciclo de execução de uma instrução serão lidos os dois registradores rs e rt do *Register File* endereçados por IR[25-21] e IR[20-16] respectivamente. Eles devem ser carregados nos registradores auxiliares A e B. A decodificação da instrução pela unidade “FSM Control” também acontece neste estado.

(estado 2) No próximo ciclo de execução a ALU deverá executar a operação indicada pela instrução usando como operandos os registradores A e B. ALUop = 10 e a ALU *control* usando o campo *function* IR[5-0] deverá gerar os 3 bits de controle da ULA apropriados.

(estado 3) Neste ciclo o conteúdo do registrador ALUout deve ser armazenado no registrador destino.