UFMG

Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

**Organização de Computadores I**

# Lab 3 : Banco de Registradores

*Professor:* Antonio Otavio Fernandes.
*Monitor:* Omar Vidal Pino.

5 de Outubro de 2016

## 1 Introdução

A arquitetura MIPS é do tipo RISC e para dar suporte ao processamento de dados, ela possui 32 registradores de 32 bits cada, alguns de uso geral e outros de uso específico. Todas as operações lógicas e aritméticas envolvem um ou dois registradores como fonte de dados e um registrador como destino. Num mesmo ciclo, o banco de registradores deve prover o conteúdo de dois registradores e receber um dado a ser atribuído a um terceiro registrador. Eles são identificados através dos campos **rs**, **rt** e **rd** do opcode de cada instrução, cada um com cinco bits, podendo ser arbitrariamente definidos (Ver Figura 1.1) . Há também um registrador especial, o de número zero, que armazena a constante 0 definida por *hardware*.


Figura 1.1: Formato de uma instrução do *tipo-R* com a definição dos campos de bits **rs**, **rt** e **rd**, relativos aos registradores da arquitetura MIPS

Um *banco de registradores* é um componente digital composto por um conjunto de registradores que podem ser acessados de forma organizada. De uma maneira geral, podem ser executadas operações de leitura dos dados anteriormente gravados e de escrita de dados para modificar as informações internas. Este componente *é um dos componentes mais importantes do fluxo de dados* em um processador. As informações que estão sendo processados em um determinado momento devem estar armazenadas no banco de registradores. A figura 1.2 abaixo ilustra o banco de registradores do processador MIPS [Patterson - Hennessy, 2011].

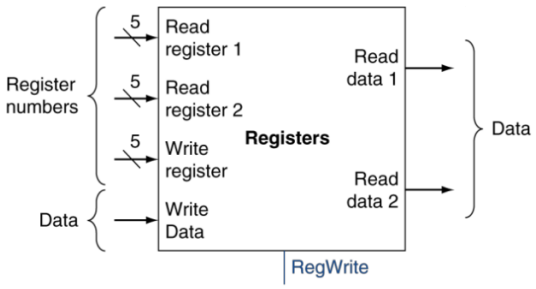


Figura 1.2: Diagrama em blocos do banco de registradores da arquitetura MIPS

Como é mostrado na figura 1.2 , o *banco de registradores* do MIPS é composto por 32 registradores de 32 bits cada, organizados em uma estrutura com três portas de acesso: duas portas de leitura *ReadData1* e *ReadData2* , que mostram dados armazenados em registradores identificados pelas entradas *ReadRegister1* e *ReadRegister2*. Esta leitura é feita de forma assíncrona, ou seja, assim que uma das entradas de endereço de leitura *ReadRegister1* ou *ReadRegister2* (ou ambas) for modificada, a respectiva saída mostrará o dado. A terceira porta *WriteData* é para gravar um novo dado no registrador identificado pela entrada *WriteRegister*. Esta escrita é realizada de forma síncrona com o acionamento da entrada de relógio *CLK* e pela habilitação da operação através do sinal *RegWrite*.

## 1.1 Organização Lógica de um Banco de Registradores

De uma forma geral, um banco de registradores pode ser organizado conforme explicado a seguir. Primeiro,temos um grupo de registradores internos ao banco de registradores. Estes registradores devem ter alguns sinais de controle em comum, como por exemplo, os sinais de relógio (clock), de carga paralela (load) e de apagamento do conteúdo (clear).

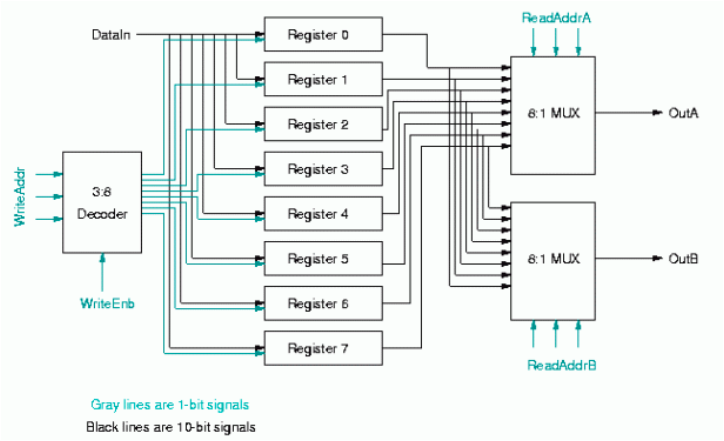


Figura 1.3: Exemplo de organização lógica de um Banco de 8 registradores.

A figura 1.3 apresenta a implementação lógica de duas portas de leitura (*OutA* e *OutB*) em um banco de 8 registradores. Cada saída ou porta de leitura é conectada em um multiplexador 8x1, cuja entrada é selecionada pelo identificador do registrador a ser lido (*ReadAddrA* e *ReadAddrB*). Todas as saídas dos 8 registradores são conectados neste multiplexador, resultando em um circuito complexo.

A implementação da porta de escrita é um pouco mais complexa, porque somente um registrador deve ter seu conteúdo atualizado. Isto pode ser realizado com auxílio de um decodificador que gera um sinal pode ser usado para determinar o registrador a ser escrito. É importante lembrar que o estado do registrador deve ser alterado somente na borda ativa do sinal de relógio.

# 2 Objetivos

O objetivo desta aula prática é a implementação do banco de registradores da arquitetura MIPS com suporte a todas as operações definidas para ele. A figura 1.2 ilustra as entradas e saídas a serem disponibilizadas. As entradas de *ReadRegister1*, *ReadRegister2* e *WriteRegister* definem os três registradores envolvidos na operação, sendo obtidos diretamente dos respectivos campos **rs**, **rt** e **rd** da instrução de 32 bits, como mostrado na figura 1.1.
As saídas *ReadData1* e *ReadData2* e a entrada *WriteData* são de 32 bits e contém os dados lidos de dois registradores e o dado a ser escrito no terceiro. O sinal de controle *RegWrite* é ativado sempre que uma operação de escrita deve atualizar o conteúdo de um registrador com o dado presente na entrada *WriteData*.
Por se tratar de lógica sequencial (lembre-se que registradores são conjuntos de flip-flops do tipo D), é necessária a adição de um sinal de relógio, não ilustrado na figura 1.2, para que o comportamento seja o de um circuito síncrono.

# 3 Parte Experimental

A parte experimental consiste em testar o projeto do *banco de registradores*, incluindo as operações de leitura e escrita arbitrárias nos registradores.

## 3.1 Atividade 1

Crie um novo projeto (module) para um Banco de registradores e salve o modulo com o nome BANCO.v. Um TEMPLATE do módulo é mostrado abaixo. (Ver pasta *code-examples*.)

```
module BANCO(readAddress1,      //  indice do registrador rs
             readAddress2,      //  indice do registrador rt
             writeAddress,      //  indice no registrador rd ou rt
             clk,               //  Relogio
             writeEn,           //  controle de escrita
             writeData,         //  entrada de dados para escrita
             readData1,         //  saida de dados do registrador rs
             readData2);        //  saida de dados do registrador rt
// Parametros
parameter REGFILE_WIDTH = 5;      // numero de bits para
                                  // enderecamento de memoria
parameter DATA_WIDTH = 32;        //
parameter REGFILE_R_WIDTH = 32;   // quantidade de registradores no Banco

//pin declarations
//INPUT
input wire writeEn, clk;
input wire [REGFILE_WIDTH-1:0] readAddress1, readAddress2, writeAddress;
input wire [DATA_WIDTH-1:0] writeData;

// OUTPUT
output wire [DATA_WIDTH-1:0] readData1, readData2;
 // ... your code
endmodule
```

## 3.2 Atividade 2

Nesta atividade, você deverá verificar o funcionamento do projeto do banco de registradores através de sua simulação. Para isto,para testar a operação do *banco de registradores*, carregue o *ModelSim* e crie um novo projeto incluindo, além de seu projeto original *BANCO.v*, o arquivo de testes *"testb.v"*(Ver pasta *code-examples*). Compile os dois arquivos e simule as operações de leitura e escrita seguindo os passos estudados na última aula.

## 3.3 Atividade 3

A atividade 3 consiste em combinar o código do Lab2 (ULA) com o código do *banco de registradores*, num novo projeto, de forma que as operações lógicas e aritméticas possam ser realizadas e armazenadas de volta nos registradores. Uma ideia de como projetar o fluxo de dados é mostrado na da figura 3.1 que inclui um banco de registradores, uma ULA e ummultiplexador de dados de 4 bits.

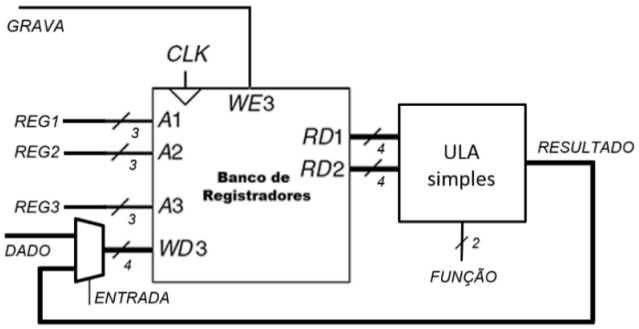


Figura 3.1: Fluxo de dados a projetar.

Utilizando apenas as operações com registradores e as funções definidas para a ULA, simule a sequência de operações necessárias (leitura de registradores seguida de operação lógica ou aritmética e escrita em registrador). Mostre exemplos de sequências de operações e os resultados correspondentes.

# 4 Relatorio

Deve ser entregue uma pasta compactada contendo os seguintes arquivos :

- Arquivo fonte e código compilado (módulos e testbench), sem erros de execução.
- Arquivos de entrada e saída para uma execução exemplo.
- Relatório com os resultados das actividades 1-3 do laboratório.

Arquivos com vírus não serão avaliados. Todos os trabalhos plagiados/copiados serão desconsiderados (zerados).