Arquitetura de Computadores

João Victor Briganti de Oliveira[1], Luiz Gustavo Takeda[2]

Universidade Tecnológica Federal do Paraná – UTFPR

COCIC – Coordenação do Curso de Bacharelado em Ciência da Computação
Campo Mourão, Paraná, Brasil

[1]joliveira.2022@alunos.utfpr.edu.br
[2]luiztakeda@alunos.utfpr.edu.br

Resumo

Especificações e escolhas dos meios para desenvolver o Banco de Registradores e a Unidade Logica de Aritmética.

1. Introdução

O presente trabalho serve como documentação para a arquitetura criada para a turma de Arquitetura e Organização de Computadores de 2023. A arquitetura sendo construída tem 32-bits, e segue o modelo da arquitetura MIPS(Microprocessor without Interlocked Pipeline Stages) [1].

A arquitetura sendo construída tem 32-bits, e segue o modelo da arquitetura MIPS(Microprocessor without Interlocked Pipeline Stages) [1]. Para a montagem da arquitetura foi utilizado o simulador Logisim [2], uma ferramenta usada para simulação de circuitos digitais.

A sessão 2 trás a implementação do banco de registradores e seus detalhes, a sessão 3 trás a implementação da ULA(Unidade Lógica Aritmética).

2. Banco de Registradores

Para os registradores o modelo do MIPS será usado, onde temos 32 registradores que são divididos entre salvos, temporários e especiais. A tabela [1] abaixo exemplifica a divisão e nomenclatura desses registradores:

Tabela 1: Lista e Descrição dos Registradores

Lista de Registradores		
Registradores	**Nome**	**Descrição**
$0	**$zero**	**Sempre zero**
$1	**$at**	**Reservado para o montador**
$2-$3	**$v0,$v1**	**Valores de retorno**
$4-$7	**$a0-$a3**	**Argumento(s) da função**
$8-$15 e $24-$25	**$t0-$t7 e $t8-$t9**	**Registradores Temporários**
$16-$23	**$s0-$s7**	**Registradores Salvos**
$26-$27	**$k0-$k1**	**Reservado para o Kernel**
$28	**$gp**	**Ponteiro Global**
$29	**$sp**	**Ponteiro de Pilha**

$30	**$fp**	**Ponteiro de Frame**
$31	**$ra**	**Endereço de Retorno**

O circuito do banco de registradores possui cinco entradas e duas saídas. Sendo elas:

- LR1: Load Register 1, entrada de 5-bits que especifica o primeiro registrador a ser lido.
- LR2: Load Register 2, entrada de 5-bits que especifica o segundo registrador a ser lido.
- WR: Write Register, entrada de 5-bits que especifica em qual registrador será escrito o WD
- WD: Write Data, entrada de 32-bits que traz os valores que serão escritos em um dos registradores.
- EW: Enable Write, bit que habilita/desabilita escrita de registrador. Em 0 desabilita e em 1 habilita.

Na implementação desse banco foram usados 31 registradores, o registrador $zero por não permitir escrita e ser sempre o valor 0, foi "desligado" diretamente. Essa escolha de usar um registrador a menos se dá pela natureza do registrador $zero, e pelo fato de que montando o circuito dessa maneira se torna possível diminuir a complexidade do mesmo economizando em espaço e transistores. Para as demais partes do circuito temos a utilização de dois multiplexadores que são usados para o envio de dados nas portas LD1 e LD2 e um demutiplexador para habilitar a escrita de dados que é enviado na porta WD. Abaixo a figura [1] que ilustra com suas entradas:

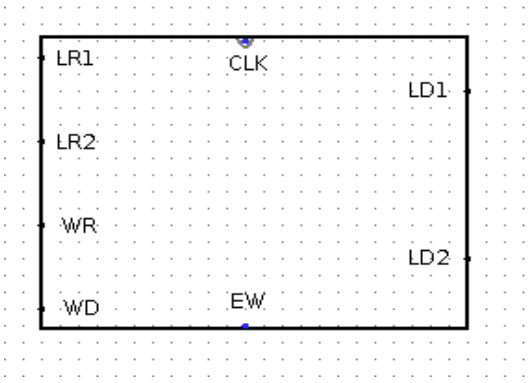

Figura 1: Banco de Registradores

Do circuito interno segue a figura[4]:

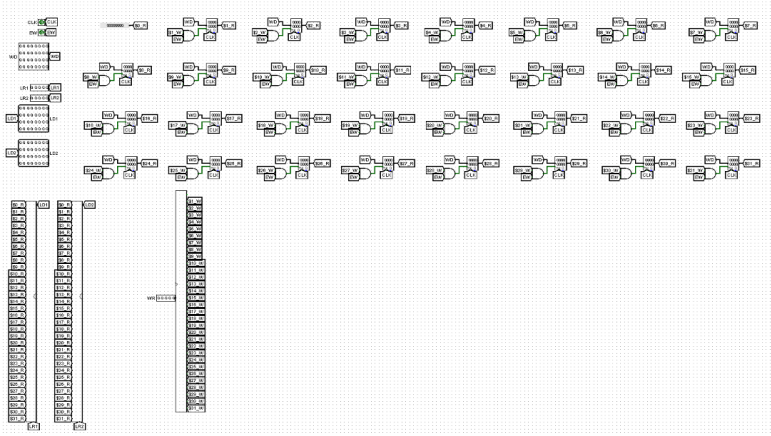

Figura 2: Circuito interno Banco de Registradores

3. Unidade Lógica Aritmética

3.1 Circuito

Para realizar as operações temos as seguintes entradas, **Operação** 4bits responsável por selecionar qual operação vai ser utilizada, **Shamt** 5bits utilizado para ditar quantas vezes será deslocado nos comandos **SLL** e **SRL**, **A** 32bits e **B** 32bits são as entradas para a operação.

E a saída, **Zero** 1bit, que indica quando o resultado é 0, **Result** 32bits é o resultado da operação e **Overflow** 1bit indica quando há um estouro de memória nas operações de **ADD** e **SUB.**

A figura[3] abaixo, ilustra a disposição das entradas e saídas:

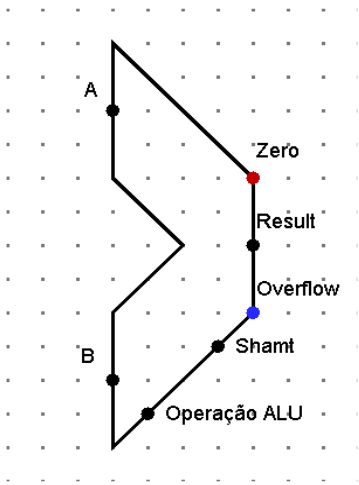

Figura 3: ULA

Do circuito interno segue a figura[4]:

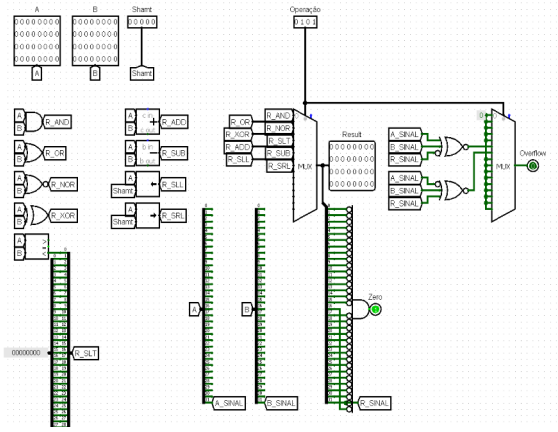

Figura 4: Circuito interno ULA

Funcionamento do circuito:

Sobre a entrada da seleção de operação é utilizado um MUX para multiplexar todos os resultados das operações em uma única saída, assim de acordo com o valor de **Operação**, o MUX direciona o valor de resultado de uma operação específica para o **Result.**

Para a saída **Overflow**, é utilizado um MUX, a fim de tratar o valor de overflow dependendo da operação que está sendo realizada, neste caso, apenas a operação de ADD e SUB, estão sendo tratas, as demais operações o O**verflow**, é constante 0. Para o caso de ADD, quando os sinais são iguais entre A e B, e o sinal do valor resultando é o inverso, tem se o overflow. Na operação SUB, quando os valores dos sinais A e B são distintos, e o sinal do resultado for igual ao sinal de B, temos overflow.

A saída **Zero**, é verificado se todos os valores de **Result**, são iguais a 0, utilizando uma porta AND com suas entradas negadas, quando caso algum valor seja 1, sua saída será 0.

3.2 Operações

As operações implementadas na ULA, são as seguintes:

AND

Utiliza a porta lógica AND: R = A & B.
Operação: 0x0
Shamt: Indiferente

OR

Utiliza a porta lógica OR: R = A | B.
Operação: 0x1
Shamt: Indiferente

NOR

Utiliza a porta lógica NOR: R = ~(A | B).
Operação: 0x2
Shamt: Indiferente

XOR

Utiliza a porta lógica XOR: R = A ^ B.
Operação: 0x3
Shamt: Indiferente

SLT

Realiza a comparação: R = A < B.
Operação: 0x4
Shamt: Indiferente

ADD

Realiza a soma: R = A + B.
Operação: 0x5
Shamt: Indiferente

SUB

Realiza a subtração: R = A - B.
Operação: 0x6
Shamt: Indiferente

SLL

Realiza o deslocamento para esquerda: R = A << Shamt .
Operação: 0x7
Shamt: Quantidade de deslocamento

SRL

Realiza o deslocamento para direita: R = A >> Shamt .
Operação: 0x8
Shamt: Quantidade de deslocamento

4. Referências

[1] D. A. Patterson and J. L. Hennessy, *Computer organization and design: The hardware/software interface*. in Morgan Kaufmann series in computer architecture and design. Morgan Kaufmann, 2013.

[2] “Logisim,” www.cburch.com. http://www.cburch.com/logisim/

[3] J. L. Hennessy and D. A. Patterson, *Computer architecture: A quantitative approach*. in The morgan kaufmann series in computer architecture and design. Elsevier Science, 2017.