

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA
Bacharelado em Engenharia de Software

**Ian Asenjo Dominguez Cunha, Laura Lourdes Coutinho Rodrigues, Warley
Leandro dos Anjos**

Trabalho Prático III: Simulador de Instruções do MIPS

Belo Horizonte
2020

**Ian Asenjo Dominguez Cunha, Laura Lourdes Coutinho Rodrigues, Warley
Leandro dos Anjos**

Trabalho Prático III: Simulador de Instruções do MIPS

Trabalho sobre Simulador de Instruções do MIPS,
apresentado como requisito parcial à aprovação na disciplina
Arquitetura de Computadores.

Professor: Pedro Henrique

**Belo Horizonte
2020**

SUMÁRIO

1. CATEGORIA

1.1. Unidade Lógica Aritmética 4

1.2. Banco de Registradores 4

1.3. Memória Principal 4

2. METODOLOGIA 5

3. CONSIDERAÇÕES FINAIS 5

REFERÊNCIAS 5

ANEXOS 6

1.Categoria

1.1 Unidade Lógica Aritmética

A Unidade Lógica Aritmética (ULA) é um componente do caminho de dados que realiza operações lógicas e aritméticas. Normalmente, a ULA recebe dois operandos como entrada, e uma entrada de controle que permite especificar qual operação deverá ser realizada.

Por esse motivo, a construção de uma ULA se baseia em dois fundamentos principais: o controle de fluxo de dados e a construção de circuitos que implementam operações.

Exemplos de operações realizadas pelo componente são lógica (AND,OR,NOT,XOR,NAND) e aritméticas (ADD, SUB, MULT,DIV).

1.2 Banco de Registradores

O componente Banco de Registradores é composto por um conjunto de registradores que são 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.As informações que estão sendo processados em um determinado momento devem estar armazenadas no banco de registradores.

1.3 Memória Principal

Sua função é armazenar informações que são ou serão manipuladas pelo sistema para que elas possam ser recuperadas quando necessário.Existem duas únicas operações possíveis que podem ser realizadas em uma memória Load (LW) e Store (SW).

2. Metodologia

Para desenvolvimento do trabalho os grupo utilizou para comunicação as ferramentas Discord e Whatsapp, o desenvolvimento foi feito pelo Visual Studio Code, dispondo do recurso Live Share. As tarefas foram organizadas pelo Trello, em que um membro ficou responsável por auxiliar e o código fonte da aplicação está disponível no GitHub.

A linguagem utilizada para o desenvolvimento do trabalho foi Java, pois todos os integrantes já estão familiarizados com suas ferramentas.

3. Considerações Finais

Este trabalho faz parte de uma série que tem como objetivo final a construção de um simulador de uma máquina MIPS.

Foi implementado um programa que simula o funcionamento de uma máquina MIPS, dotada de Unidade Lógica e Aritmética, Banco de Registradores e Memória Principal.

Referências

https://www2.pcs.usp.br/~labdig/pdffiles_2014/banco-registradores.pdf

<http://www.dca.fee.unicamp.br/~tavares/courses/2015s2/ea773-3.pdf>

https://www.ic.unicamp.br/~pannain/mc722/aulas/arq_hp5.pdf

http://www.ic.uff.br/~boeres/slides_FAC/FAC-aula3.pdf

Anexos

<https://github.com/WarleyLeandro/simulador-de-instrucoes-MIPS>

Figura 1: Memória

```
1  import java.util.ArrayList;
2
3  public class Memory {
4
5      public String endereco;
6      public String instrucao;
7      public String dados;
8      ArrayList<Memory> memory = new ArrayList<Memory>();
9
10     // salva o dado e instrução na memória
11     public void salvaDados(String endereco, String instrucao, String dados) {
12         Memory men = new Memory();
13         if(instrucao == " ") {
14             men.instrucao = null;
15             men.endereco = endereco;
16             men.dados = dados;
17             memory.add(men);
18         } else if(dados == " ") {
19             men.endereco = endereco;
20             men.instrucao = instrucao;
21             men.dados = null;
22             memory.add(men);
23         } else {
24             System.out.println("Dados inválidos!");
25         }
26     }
27
28     // carrega a instrução da memória
29     public String carregaInstrucao(String buscaEnd) {
30         for (int i = 0; i < memory.size(); i++) {
31             Memory temp = memory.get(i);
32             if (temp.endereco == buscaEnd) {
33                 return temp.instrucao;
34             }
35         }
36         return "Instrução não encontrada na memória";
37     }
38
39     // carrega o dado da memória
40     public String carregaDados(String buscaEnd) {
41         for (int i = 0; i < memory.size(); i++) {
42             Memory temp = memory.get(i);
43             if (temp.endereco == buscaEnd) {
44                 return temp.dados;
45             }
46         }
47         return "Dado não encontrado na memória";
48     }
49 }
```

Figura 2: Memória Continuação

```
51 // gera todos endereços de 32
52 public String geraEndereco(int indice) {
53     String[] enderecos = { "0x000", "0x001", "0x002", "0x003", "0x004", "0x005", "0x006", "0x007", "0x008", "0x009",
54         "0x010", "0x011", "0x012", "0x013", "0x014", "0x015", "0x016", "0x017", "0x018", "0x019", "0x020",
55         "0x021", "0x022", "0x023", "0x024", "0x025", "0x026", "0x027", "0x028", "0x029", "0x030", "0x031" };
56     return enderecos[indice];
57 }
58
59 // imprimir arquivo
60 public ArrayList<Memory> imprimeMemoria() {
61     return memory;
62 }
63
64 }
```

Figura 3: ALU

```
1 public class ALU {
2     public static final short ADD = 0;
3     public static final short SUB = 1;
4     public static final short MULT = 2;
5     public static final short DIV = 3;
6     public static final short AND = 4;
7     public static final short OR = 5;
8     public static final short XOR = 6;
9     public static final short NOR = 7;
10    public static final short SLT = 8;
11    public static final short SLL = 9;
12    public static final short SRL = 11;
13    public static final short JR = 12;
14    public static final short ADDI = 13;
15    public static final short ANDI = 14;
16    public static final short ORI = 15;
17    public static final short SLTI = 16;
18    public static final short BEQ = 17;
19    public static final short LW = 18;
20    public static final short SW = 19;
21    public static final short J = 20;
22    public static final short JAL = 21;
23
24    public String getInstrucao(int indice) {
25        String[] name = { "ADD", "SUB", "MULT", "DIV", "AND", "OR", "XOR", "NOR", "SLT", "SLL", "SRL", "JR", "ADDI", "ANDI", "ORI", "SLTI", "BEQ", "LW", "SW", "J", "JAL" };
26        return name[indice];
27    }
28
29 }
```

Figura 5: Trello da Equipe

